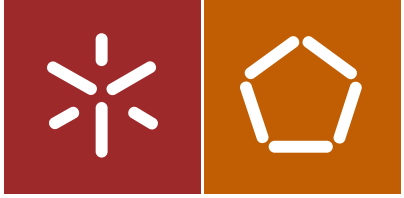
Universidade do Minho
Escola de Engenharia

João Filipe Cruz Sousa

Scanner a 3D

João Filipe Cruz Sousa Scanner a 3D
UMinho | 2015

janeiro de 2015



Universidade do Minho
Escola de Engenharia

João Filipe Cruz Sousa

Scanner a 3D

Dissertação de Mestrado
Ciclo de Estudos Integrados Conducentes ao Grau de
Mestre em Engenharia Eletrónica Industrial e Computadores

Trabalho efetuado sob a orientação do
Professor Doutor António Fernando Macedo Ribeiro
Professor Doutor Paulo José de Albuquerque Cardoso Trigueiros

janeiro de 2015

DECLARAÇÃO

Nome: João Filipe Cruz Sousa
Endereço eletrónico: j_sousa11@hotmail.com Telefone: 914957162 / 252613141
Bilhete de Identidade/Cartão do Cidadão: 13839281
Título da dissertação: SCANNER A 3D

Orientadores:
Professor Doutor António Fernando Macedo Ribeiro
Professor Doutor Paulo José de Albuquerque Cardoso Trigueiros

Ano de conclusão: 2015
Mestrado em Engenharia Eletrónica Industrial e Computadores

Universidade do Minho, _____/_____/_____

Assinatura:

"Obstacles don't have to stop you. If you run into a wall, don't turn around and give up. Figure out how to climb it, go through it, or work around it."

Michael Jordan

AGRADECIMENTOS

Um desafio tão grande quanto escrever a tese foi arranjar palavras para agradecer a todas as pessoas que de uma forma ou de outra me ajudaram a completar este percurso da minha vida. Ao longo destes últimos meses foram inúmeros os momentos de frustração e desânimo que se apoderaram de mim, e foram nesses momentos que o incentivo e confiança da vossa parte para não desistir me deu ânimo para continuar em frente. Estas palavras, embora poucas e simples, demonstram o meu profundo agradecimento a todos vocês.

Ao *Professor Doutor Fernando Ribeiro*, expresso o meu sincero agradecimento pela orientação, apoio, e pelas diversas sugestões e criticas que me conduziram no melhor caminho. Mas acima de tudo, agradeço a oportunidade de trabalhar neste projeto. Ao *Professor Doutor Paulo Trigueiros*, o meu mais profundo agradecimento por me dar a conhecer os materiais disponíveis e por me ter proporcionado as condições necessárias para a elaboração da dissertação. Agradeço também a sua simpatia e disponibilidade em todos os momentos.

Aos técnicos das oficinas do *Departamento de Eletrónica Industrial*, *D. Ângela*, *Sr. Carlos e Sr. Joel*, pela disponibilidade e paciência no auxílio dos diversos problemas que me foram surgindo ao longo destes meses.

Aos meus colegas de curso e amigos, agradeço pela vossa amizade, companheirismo e ajuda, fatores muito importantes não só na realização deste projeto como em todos os momentos do meu percurso académico neste últimos anos. O meu sentido pedido de desculpas por em diversos momentos serem colocados em segundo plano.

À minha recente família, os meus afilhados e pseudo afilhada, pelo apoio incondicional em todos os momentos. Agradeço por me proporcionarem os melhores momentos no meu percurso académico e por nunca terem desistido de me fazer acreditar que era possível.

Um especial agradecimento os meus pais por todos os sacrifícios e esforço que fizeram ao longo da sua vida para me darem o melhor possível e para proporcionarem os meios para eu ter chegado até aqui.

Por último à *Joana*, a minha companheira de percurso vivencial, pelo inestimável apoio que sempre me deu, pela sua bondade e por tudo o que representa para mim.

RESUMO

Ao longo dos anos, a diminuição do custo dos equipamentos de digitalização *3D* levou à sua utilização em diversas novas áreas. Contudo, as soluções existentes no mercado para *scanners* a *3D* ainda não são muito práticas nem muito acessíveis financeiramente

Em *2010*, com o lançamento do sensor *Kinect* pela *Microsoft* foram criadas um conjunto de novas oportunidades na área da computação e visão por computador. Os sensores de profundidade de baixo custo são uma alternativa atraente aos *scanners* a laser. Devido ao seu hardware simples e económico este dispositivo tem vindo a emergir no mercado como uma solução de mapeamento e digitalização *3D*.

A principal motivação deste trabalho é avaliar a utilização do sensor *Microsoft Kinect* na reconstrução de modelos tridimensionais de pequenos objetos. A utilização de uma plataforma rotativa permite facilitar o processo de captura da informação *3D* de um objeto. A biblioteca *Point Cloud Library* (*PCL*) tem como objetivo fornecer suporte para todos os blocos de construção que uma aplicação *3D* necessita, a partir dos dados das nuvens de pontos obtidos por dispositivos de perceção tridimensional.

Como resultado da implementação deste trabalho é possível obter diversas nuvens de pontos de um determinado objeto posicionado numa superfície plana, filtrar e segmentar a informação do ambiente que o rodeia e aplicar um processo de registo de nuvens de pontos capaz de devolver um modelo global desse objeto. O modelo resultante é suavizado por forma a remover alguns erros inerentes à captura por este sensor e reconstruído numa malha tridimensional para que seja posteriormente utilizado num programa de *CAD*.

Os resultados indicam que apesar da baixa resolução e da natureza bastante ruidosa do sensor *Kinect*, a qualidade dos modelos *3D* dos objetos é satisfatória. O processo de registo implementado não consegue resolver a acumulação de erro de transformação entre as sucessivas nuvens de pontos causando a falha na operação de construção do modelo global. Assim sendo, o registo de nuvens de pontos num único modelo não foi conseguido com sucesso, embora tenham sido encontrados bons resultados para cada um dos algoritmos implementados nesta aplicação. Dessa forma, as malhas tridimensionais obtidas a partir dos modelos parciais dos objetos sugerem resultados razoáveis.

Palavras-Chave: *Scanner 3D, Microsoft Kinect, Point Cloud Library*

ABSTRACT

Over the years, *3D* scanning devices cost reduction has led to its use in several new areas. However, existing solutions in the market for *3D* scanners are still not very practical or affordable financially.

In 2010, with *Kinect* sensor release from *Microsoft* a set of new opportunities in the field of computing and computer vision were created. Low-cost depth sensors are an attractive alternative to laser scanners. Due to its simple and inexpensive hardware, this device has emerged in the market as a mapping and *3D* scanning solution.

The main motivation of this work is to evaluate the use of the *Microsoft Kinect* sensor in the three-dimensional models reconstruction of small objects. The use of a rotating platform facilitates the *3D* information capturing process of an object. The *Point Cloud Library* (*PCL*) aims to provide support for all the building blocks that a *3D* application needs, from the point clouds data obtained by three-dimensional perception devices.

As a result of this work implementation is possible to obtain different point clouds of a particular object positioned on a flat surface, to filter and to segment information of the environment that surrounds it and to apply a point cloud registration process capable of returning a global model of that object. The resulting model is smoothed in order to remove some errors inherent to the capture by this sensor and rebuilt in a three-dimensional mesh to be further used in a *CAD* program.

The results show that in despite of the low resolution and very noisy nature of *Kinect* sensor, the quality of the *3D* models of the objects is satisfactory. The implemented registration process can't resolve the accumulation of processing error between successive point clouds failing the global model building operation. Therefore, the point cloud into a single register has not been successfully achieved, although good results have been found for each of the algorithms implemented in this application. Thus, the three-dimensional meshes obtained from the objects partial models suggest reasonable results.

Keywords: *3D Scanner, Microsoft Kinect, Point Cloud Library*

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

ABREVIATURAS

2D	*Espaço bidimensional*
3D	*Espaço tridimensional*
CAD	*Computer Aided Design*
CCD	*Charge-coupled Device*
CMM	*Coordinate measuring machine*
CNC	*Comando numérico computadorizado*
DoG	*Difference of Gaussian*
FLANN	*Fast Library for Approximate Nearest Neighbor*
FPFH	*Fast Point Feature Histogram*
ICP	*Iterative Closest Point*
OpenNI	*Open Natural Interaction*
PCD	*Point Cloud Data*
PCL	*Point Cloud Library*
PFH	*Point Feature Histogram*
RANSAC	*Random Sample Consensus*
RGB	*Espaço de cores Red, Green, Blue*
ROS	*Robot Operating System*
SIFT	*Scalar Invariant Feature Transform*
SPFH	*Simplified Point Feature Histogram*
TOF	*Time of Flight*
VTK	*Visualization ToolKit*

1. INTRODUÇÃO

Este capítulo apresenta uma visão geral dos tópicos incluídos nesta dissertação, bem como a sua motivação. O grande objetivo do projeto é estabelecer um scanner 3D simples e fácil de utilizar, recorrendo a baixos requisitos de hardware. Contudo, é possível perceber a divisão deste principal objetivo em diversas etapas por forma a solucionar este problema. Por último, é feita uma descrição da estrutura e organização deste documento.

1.1 Motivação

Os métodos de fabrico tradicionais envolvem uma grande quantidade de esforço, despesas e tempo. Os técnicos necessitam muitas vezes de criar moldes individuais, montar vários componentes e construir itens de várias peças. Este processo pode envolver diversos materiais, uma grande variedade de profissionais altamente treinados e um elevado número de testes bastante dispendiosos antes que o objeto perfeito seja finalmente criado.

A prototipagem rápida e a impressão *3D* veio reduzir o custo do processo de criação de novos objetos. Para além de ser um método mais rápido, a prototipagem rápida permite um maior controlo sobre a forma final do que qualquer outro método de construção. Através de técnicas de impressão avançadas é permitido imitar com bastante precisão a aparência e as características dos protótipos dos produtos em questão. Em vez de se utilizar uma fábrica para criar uma amostra de testes, em impressoras independentes podem-se criar objetos a partir de um modelo tridimensional. Estes modelos podem ser concebidos por programas de modelação ou obtidos através de *scanners 3D*.

Nos últimos anos, com a banalização de sistemas de prototipagem rápida, o preço das impressoras *3D* tornaram-se mais acessíveis para utilizadores domésticos, assim como para pequenas e médias empresas, levando a prototipagem da indústria para o local de trabalho. Foi nesse sentido que surgiu uma necessidade cada vez maior de utilização de programas de modelação, mais conhecidos como softwares de *CAD*, por parte dos utilizadores domésticos. Contudo, a utilização destes programas ainda não é tão fácil como parece. Para contornar este problema, o próximo passo será a digitalização de determinada forma tridimensional já existente para inclusão num software de *CAD*. Para proceder à digitalização de um determinado objeto utiliza-se um *scanner 3D*.

Um *scanner 3D* é um dispositivo que analisa um objeto do mundo real para recolher dados sobre a sua forma e, possivelmente, a sua aparência. Estes dados após serem recolhidos podem então ser utilizados para construir modelos tridimensionais.

Durante muitos anos, a digitalização *3D* foi amplamente utilizada para aplicações industriais, tais como engenharia inversa e inspeção de peças. Ao longo dos últimos anos, a diminuição dramática no custo dos equipamentos de digitalização *3D* levou à sua utilização crescente para muitas outras aplicações, incluindo prototipagem rápida e modelação, arte e arquitetura, educação e pesquisa, computação gráfica e efeitos visuais, realidade virtual, preservação do património, entre outros. Mais recentemente no campo biomédico, é utilizada para reconstrução de partes anatómicas, planeamento do tratamento ortodôntico, pesquisa de deformação craniana e estudo morfológico da cartilagem.

Atualmente, já existem no mercado diversas soluções para digitalização *3D*, no entanto estas ainda não são muito práticas nem muito acessíveis financeiramente. Existe uma grande variedade de técnicas para a aquisição de modelos tridimensionais de objetos, todos com uma ampla gama de custo de hardware, e diferentes níveis de exatidão e detalhe nos modelos geométricos capturados. Técnicas baseadas em aquisição de imagem e vídeo, métodos de obtenção da forma dos objetos utilizando lasers e luzes estruturadas, são alguns dos métodos que têm sido estudados nos últimos anos. Existe um enorme potencial para expandir o uso de modelos tridimensionais ainda mais, continuando a desenvolver sistemas mais simples e de custo reduzido para aquisição das características da forma externa de objetos arbitrários.

Os avanços recentes no desenvolvimento das câmaras de profundidade *3D*, como por exemplo o sensor *Microsoft Kinect*, criou um conjunto de novas oportunidades na área de digitalização *3D*. Os sensores de profundidade de baixo custo são uma alternativa atraente em relação a *scanners* a laser dispendiosos em áreas de aplicação, tais como mapeamento de espaços interiores, vigilância, robótica, entre outros. Desde o seu anúncio em 2009, o sensor *Kinect* tem vindo a causar grande expetativa na comunidade académica de computação gráfica e visão por computador.

Neste projeto de dissertação, um sistema de digitalização *3D* baseado no sensor *Microsoft Kinect* é apresentado. A avaliação da utilização deste sensor para a reconstrução de modelos tridimensionais é a principal motivação pessoal para o desenvolvimento deste projeto. As características deste sensor torna-o adequado para a sua utilização na reconstrução *3D*, e como tal por forma a solucionar este problema de investigação.

1.2 Objetivos

O foco de pesquisa desta dissertação é o desenvolvimento de um *scanner 3D* para leitura da forma de um objeto de pequenas dimensões, a fim de disponibilizar esta mesma forma tridimensional num formato de *CAD*, para posteriormente poder ser editada pelo utilizador num programa de *CAD*. Para tal, este projeto de dissertação pode ser definido por objetivos parciais.

O primeiro objetivo passa por estudar a melhor tecnologia a utilizar na implementação deste projeto. As tecnologias possíveis de ser implementadas podem ser variadas, desde a utilização de soluções de contacto ou de não contacto. No caso das soluções sem contacto, é possível ainda recorrer a diversas técnicas, desde a utilização de luz laser, raios-x, ultra-sons, ou através de câmaras. Com esta pesquisa, é esperado que a tecnologia escolhida seja prática e acessível financeiramente no sentido de reduzir o custo de produção dos atuais *scanners 3D*.

O segundo objetivo desta dissertação é a construção do hardware necessário para assegurar o funcionamento da tecnologia escolhida.

O terceiro objetivo consiste no desenvolvimento do software capaz de adquirir e processar os dados obtidos, de modo a ser possível construir a forma tridimensional do objeto.

Por último, pretende-se fazer o desenvolvimento de leitura da forma tridimensional com geração de um ficheiro passível de ser importado num programa de *CAD*.

1.3 Estrutura do documento

Este documento está divido em oito capítulos. Neste capítulo, é inicialmente feita uma breve contextualização ao projeto da presente dissertação.

Para além deste capítulo, no qual é ainda descrita a motivação e os objetivos desta dissertação, no segundo capítulo é apresentada uma revisão bibliográfica das tecnologias passíveis de serem utilizadas para implementação de um *scanner 3D* e algumas aplicações existentes no mercado das diversas tecnologias.

No terceiro capítulo, são analisadas de forma detalhada as características da tecnologia escolhida, assim como as bibliotecas e algoritmos utilizados para a aquisição e processamento dos dados obtidos por essa tecnologia.

De seguida, no quarto capítulo, é descrita pormenorizadamente o método utilizado para implementação dos algoritmos abordados no capítulo anterior, justificando a escolha dos mesmos e, demonstrando a par e passo o seu funcionamento neste sistema.

No quinto capítulo, é apresentado todo o hardware desenvolvimento para assegurar o funcionamento da tecnologia implementada nesta dissertação.

No sexto capítulo, são descritas as principais caraterísticas da aplicação desenvolvida neste projeto de investigação, sendo ilustrado em algumas imagens o funcionamento deste sistema de digitalização *3D*.

No sétimo capítulo, são apresentados todos os resultados obtidos após os diversos testes e experiências realizados nos diversos algoritmos implementados neste sistema. Em cada etapa, será feita uma discussão dos resultados e, por fim uma análise global da aplicação desenvolvida.

No último capítulo, são discutidos os resultados experimentais deste projeto e referidos os aspetos que podem ser aperfeiçoados no futuro.

2. ESTADO DA ARTE

O objetivo do presente capítulo é providenciar uma visão geral das diversas técnicas disponíveis atualmente para a digitalização 3D. É realizado um estudo detalhado do funcionamento de cada uma delas, evidenciando as suas vantagens e desvantagens. Por fim, é apresentado alguns dos scanners 3D existentes que aplicam as tecnologias abordadas.

2.1 Introdução

As tecnologias de digitalização *3D* são utilizadas para ajudar as pessoas a obter modelos tridimensionais a partir da uma cena do mundo real. Nos dias de hoje, a aplicabilidade dos *scanners 3D* são inúmeras, desde arte e arquitetura, design, controlo de qualidade, medicina, multimédia e indústria cinematográfica, entre outros. Portanto, a utilização de modelos *3D* nas mais distintas áreas tem um enorme potencial de expansão, e como tal têm sido propostas diferentes técnicas para medir a dimensão e forma de um objeto. A fim de comparar as diversas técnicas de digitalização 3D existentes uma série de considerações importantes têm de ser feitas. Alguns dos aspetos seguintes podem ser utilizados para diferenciar os diversos sistemas:

- Custo;
- Material do objeto a ser digitalizado;
- Tamanho do objeto digitalizado;
- Portabilidade do equipamento;
- Precisão do sistema;
- Aquisição da textura;
- Produtividade da técnica;
- Habilidade exigida para a utilização do equipamento;
- Conformidade dos dados produzidos com padrões ou normas.

No entanto, na prática, não há nenhum sistema de digitalização que consiga responder da mesma forma a todas estas características acima referidas. Normalmente, um método de digitalização *3D* é especializado unicamente para um determinado tipo de aplicação.

2.2 Tecnologias de Digitalização 3D

Existe uma grande variedade de tecnologias para aquisição da forma tridimensional de um objeto. Estas tecnologias podem dividir-se em dois tipos: contacto e sem contacto. As soluções sem contacto podem ainda ser divididas em duas categorias, ativas e passivas. De seguida, são apresentadas algumas destas tecnologias, sendo descrita de uma forma detalhada o seu funcionamento, e salientando algumas vantagens e desvantagens para cada uma delas.

2.3 Contacto

2.3.1 Coordinate Measuring Machine (CMM)

Um exemplo de um *scanner 3D* por contacto é a *Coordinate measuring machine* (*CMM*). Este equipamento é constituído por um braço ou outro tipo de transporte mecânico móvel com uma sonda na extremidade que se desloca ao longo da superfície do objeto calculando as coordenadas dos pontos. Os pontos obtidos são enviados para um computador, onde são posteriormente analisados usando um software de modelação (por exemplo *CAD*) e algoritmos de regressão para um maior desenvolvimento. É uma máquina utilizada essencialmente na indústria para o controlo de qualidade das peças (*Figura 1*), sendo uma tecnologia bastante precisa. A desvantagem desta tecnologia é o facto de ser necessário contacto com o objeto que será digitalizado, o que pode resultar numa danificação ou modificação do objeto durante a operação. No caso de objetos frágeis ou de valor, como artefactos históricos, esta técnica não será a mais adequada. Para além disso, o movimento do braço onde a sonda está montada é bastante lento, sendo o funcionamento nas *CMM's* mais rápidas com poucas centenas de *hertz*, o que é relativamente baixo comparando com outras tecnologias [1]–[3].

Figura 1: Utilização da *CMM* para inspeção da qualidade da peça [4]

2.3.2 Braços articulados

Um outro exemplo de um *scanner 3D* por contacto são os braços articulados. Este equipamento é constituído por um braço robótico articulado, com braços rígidos e diversos sensores angulares para facilitar o movimento do mesmo. A digitalização é realizada através do toque da sonda, colocada na extremidade do braço robótico, nos diversos pontos da superfície do objeto (*Figura 2*). O braço mecânico que se move sobre o objeto a ser digitalizado, possibilita ao usuário capturar os dados de forma rápida e com um alto grau de resolução. Esta solução é muito utilizada na indústria de animação por computador para digitalizar modelos de barro. É uma técnica bastante precisa, no entanto menos preciso do que as *CMM's*. Esta tecnologia é bastante idêntica à *CMM*, a única diferença reside no facto de ser preciso controlar todo a operação por parte de um utilizador. Assim como nas *CMM's*, a desvantagem baseiam-se no facto de ser um processo muito lento [1], [3], [5].

Figura 2: Sistema de braço articulado [6]

2.4 Sem Contacto Ativo

Este método de digitalização, como o próprio nome indica, não têm qualquer tipo de contacto físico com o objeto a ser digitalizado. Os *scanners* sem contacto ativos emitem um determinado tipo de radiação, sendo possivelmente ultra-som, raios-x ou luz, e detetam a sua reflexão a fim de investigar um objeto ou ambiente. Algumas técnicas deste tipo incluem triangulação a laser, *time-of-flight* (*TOF*) e luz estruturada. Cada uma destas técnicas são utilizadas em diversas áreas, no entanto o princípio de funcionamento é o mesmo.

2.4.1 Triangulação a laser

O *National Research Council of Canada* [7] foi um dos primeiros institutos a desenvolver a tecnologia de digitalização por triangulação a laser em 1978 [8]. Esta técnica consiste na projeção de uma luz laser, possivelmente um padrão sob a forma de um ponto ou linha, sobre a superfície do objeto que se pretende digitalizar. Na maioria dos casos, é utilizado uma linha como padrão para acelerar o processo de aquisição dos dados. Um sensor, frequentemente uma câmara *CCD*, deteta em cada leitura a luz refletida pelo objeto e grava a sua posição exata [9]. Um exemplo de uma configuração de um sistema de triangulação é apresentado na *Figura 3*.

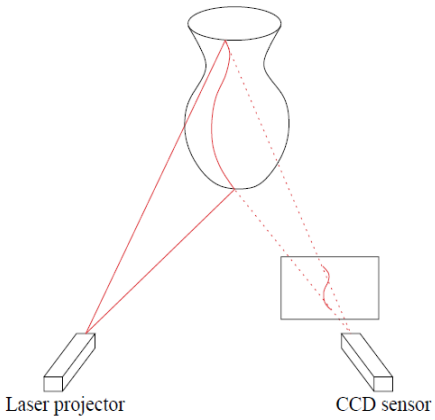


Figura 3*:* Sistema de triangulação a laser [9]

Esta técnica é designada por triangulação a laser porque o padrão do laser, a câmara *CCD* e o emissor laser formam um triângulo entre si. O comprimento de um dos lados do triângulo é conhecido, isto é, a distância entre a câmara e o emissor laser, frequentemente denominada base. O ângulo (β) entre o emissor laser e a base também é conhecido. Pela observação da posição do padrão do laser no campo de visão da câmara, o ângulo (α) entre a câmara e a base pode ser determinado. Através destas três informações é possível determinar a forma e o tamanho do triângulo. A posição da luz refletida na superfície do objeto (*p1*) pode ser encontrada a partir do triângulo resultante (*Figura 4*) [10], [11].

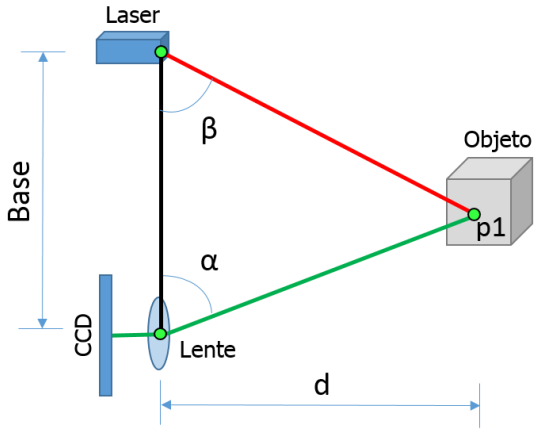


Figura 4: Método de triangulação a laser

Esta técnica pode contar com uma configuração simples ou dupla câmara (*Figura 5*). A solução com dupla câmara, com dois pontos de vista distintos, é utilizada em inúmeros projetos para diminuir os problemas de oclusão. A oclusão corresponde a uma determinada área na qual não se consegue obter nenhuma informação. Neste caso, se a câmara e o laser não estiverem necessariamente alinhados pode ocorrer uma situação em que alguns pontos da superfície analisada podem ser vistos pela câmara mas não pelo laser. Este problema depende essencialmente da configuração do sensor. No entanto, esta solução aumenta a complexidade e o preço do dispositivo, porque é necessário ter duas câmaras e em detrimento disso é indispensável um maior custo computacional para processar esta elevada quantidade de dados. Além disso, esta complexidade de hardware adicional reduziria a fiabilidade global do sistema [12].

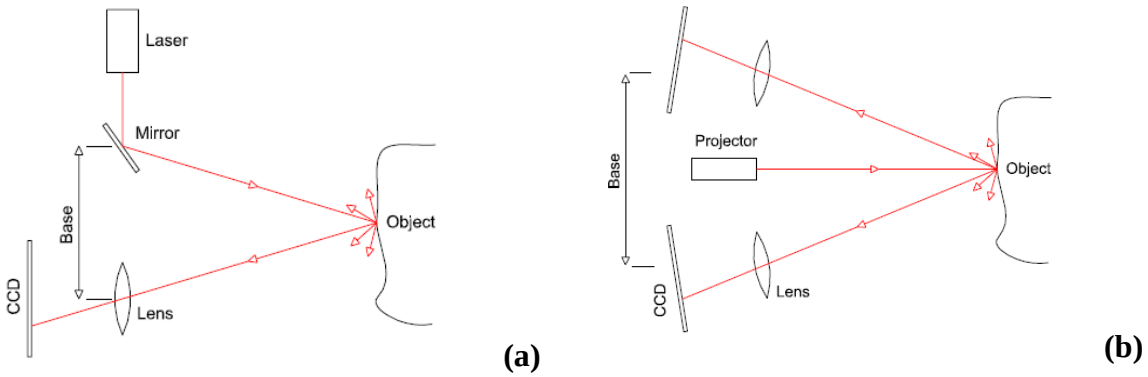


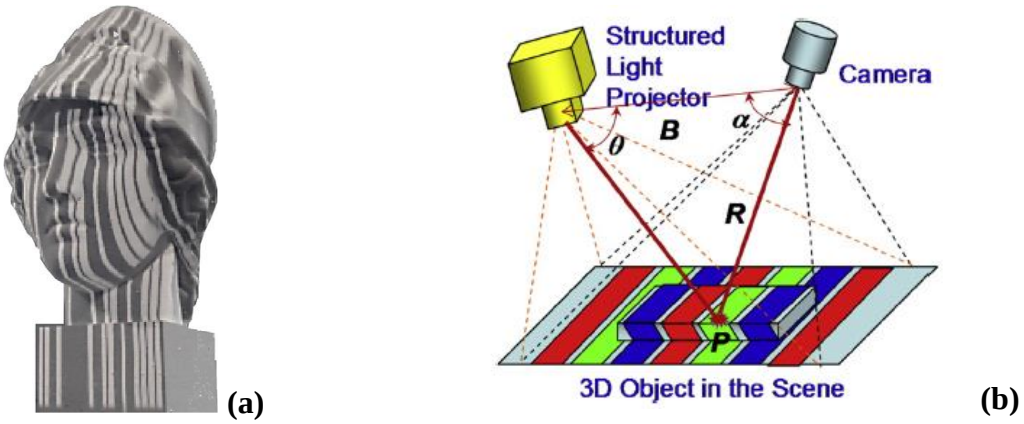
Figura 5: Triangulação a laser: solução única câmara (a) solução dupla câmara (b) [13]

Este tipo de *scanners* geralmente oferecem uma elevada resolução (inferior a 0,1 mm) a curtas distâncias. São capazes de obter os dados a grandes velocidades (superiores a 100.000 pontos por segundo), sendo um excelente método de aquisição de imagens tridimensionais de alta precisão, como por exemplo artefactos históricos e detalhes arquitetónicos. A desvantagem desta técnica baseia-se no facto da enorme dependência dos níveis de luminosidade que devem situar-se dentro de uma faixa muito específica. Outra limitação destes sistemas é a elevada dependência dos materiais da superfície que se pretende digitalizar. Se a superfície for muito brilhante ou refletiva sob condições adversas de luminosidade resulta em buracos no conjunto de dados [14].

Os *scanners* por triangulação a laser são um meio cada vez mais comuns para a aquisição da geometria tridimensional de objetos. A popularidade deste método é derivado em grande parte pela relativa robustez e precisão que é atingível. Conceitualmente, o design do scanner é muito simples, empregando geometria simples e usando as peças disponíveis normalmente. A precisão que um *scanner* a laser pode atingir está diretamente relacionado com o custo dos seus componentes. Uma elevada precisão pode ser obtida pela utilização de componentes de maior preço. Portanto, para otimizar todo este processo, a complexidade do dispositivo de digitalização tridimensional depende da aplicação para o qual é destinado. Apesar da sua simples geometria e componentes, este dispositivo deve ser projetado e calibrado com uma precisão extremamente elevada. Alguns *scanners* existentes empregam tipicamente certos componentes difíceis de calibrar. A melhor conceção pode ser encontrada através da eliminação destas partes, e a escolha de componente que são de baixo custo e fácil de calibrar [15].

2.4.2 Luz estruturada

A técnica de luz estruturada (*Structured light*) é um dos vários métodos sem contacto ativo para aquisição das coordenadas *3D* de um objeto ou ambiente. Este método consiste na projeção de um padrão específico (ou um conjunto de padrões) na superfície do objeto e na extração da informação dessa geometria através da deformação desse padrão (*Figura 6-a***)**. Em muitos casos, este método é confundido com as técnicas de digitalização por triangulação a laser, no entanto embora também se baseie em triangulação, não necessita de utilizar emissores de luz laser [16], [17].



(a) **(b)**

Figura 6: Projeção de um padrão sobre o objeto [16] (a) Sistema de luz estruturada [17] (b)

O sistema de luz estruturada é projetado em torno de dois dispositivos eletrónicos muito comuns: um projetor e uma câmara (*Figura 6-b***)**. O projetor é utilizado para fazer a projeção de um específico padrão de luz pré-definido que cobre a totalidade ou parte da superfície do objeto. Este padrão pode ser um simples conjunto de listras de diversas cores ou um complexo padrão com curvas, codificado tanto em tempo como em espaço. Posteriormente, a cena é capturada por um detetor de imagem digital, por exemplo uma câmara digital. A partir de uma imagem (ou uma sequência de imagens) adquirida pela câmara um código de listras pode ser extraído. A codificação pode ser feita usando um único padrão ou temporariamente um série de padrões variados que são projetados de forma sequencial na cena. De seguida, a imagem é processada a fim de deduzir a geometria do objeto através das deformações do padrão na imagem digital. Na maioria dos casos a informação da profundidade é reconstruída por triangulação, a partir da posição relativa do par emissor-sensor [16], [18].

A chave para estes sistemas é a técnica utilizada para diferenciar um único ponto de luz projetado a partir da imagem adquirida sob um padrão de projeção *2D*. Várias técnicas têm sido propostas nesse sentido. Estas técnicas podem ser classificadas em sequenciais (projeção de múltiplos padrões, *multi-shot*) ou por um único padrão (*single-shot*). Se o objeto alvo for estático e a aplicação não necessitar de uma restrição rigorosa do tempo de aquisição, as técnicas de projeção de múltiplos padrões (*multi-shot*) aparentam ser as mais adequadas, e pode muitas vezes originar resultados mais confiáveis e precisos [17].

A maioria dos *scanners* de luz estruturada projetam padrões binários compostos por listras pretas e brancas de forma sequencial (técnica *multi-shot*) sobre a cena que se pretende digitalizar. Estes padrões binários podem ser subdivididos até a resolução do projetor ser atingida. O código binário tem uma grande propriedade que só muda um único bit por vez. Para um determinado píxel, o respetivo código é geralmente formado pela sequência de valores de

luminância para aquele píxel em todos os padrões simulados. Um código de *Gray* está associado a cada um dos planos de luz, e cada padrão de iluminação representa 1 dos bits deste código. Este código é utilizado para reduzir erros na descodificação das imagens capturadas [19]–[21].

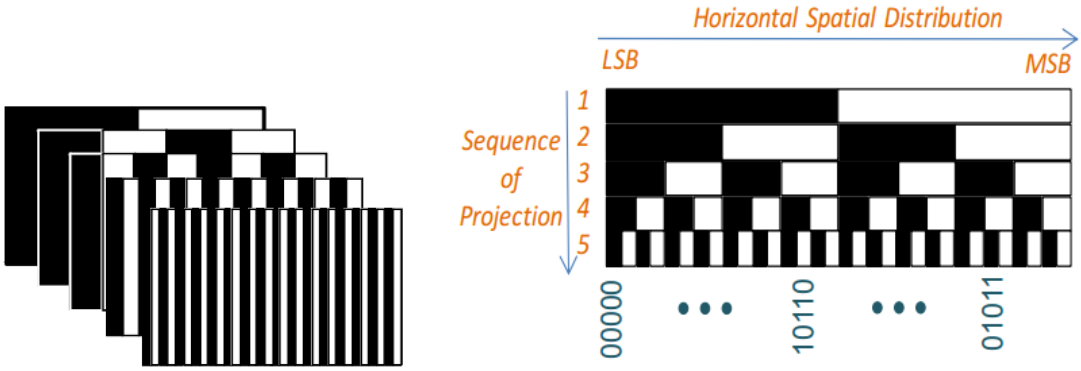


Figura 7: Projeção sequencial de padrões binários para imagens *3D* [22]

Cada ponto na superfície do objeto corresponde um código binário único que difere dos outros códigos dos diferentes pontos. Em geral, N padrões podem codificar 2^N listras. A *Figura 7* apresenta uma projeção simplificada de um padrão de 5 bits. Após esta sequência de padrões ter sido projetada num objeto estático, existem 32 ($= 2^5$) áreas únicas codificadas com listras únicas. As coordenadas *3D* podem ser computorizadas, baseadas no princípio de triangulação, para todos os 32 pontos ao longo de cada linha horizontal, formando um modelo completo de uma imagem *3D* [17].

A grande vantagem dos *scanners* de luz estruturada é a velocidade. Em vez de se digitalizar um ponto de cada vez, estes *scanners* conseguem digitalizar um múltiplo número de pontos ou a totalidade do campo de visão de uma só vez. Isto reduz ou elimina o problema de distorção do movimento. Contudo, alguns sistemas existentes são capazes de digitalizar objetos em movimento em tempo real. Apesar disso, este método é adequado para a digitalização de pequenos objetos estáticos, de modo a obter-se o resultado de forma rápida [10].

No entanto, quando é necessário digitalizar uma grande imagem ou objeto inteiro este sistema não é o mais apropriado, sendo espectável uma perda a precisão e exatidão. A digitalização de objetos largos gera uma enorme quantidade de dados redundantes, que requer um tempo extra para a aquisição dos dados e um elevado custo computacional no processamento dos mesmos. Devido à resolução das câmaras e à influência da luz ambiente, os *scanners* a laser têm um melhor desempenho no caso da digitalização de objetos de grandes dimensões [10], [23].

Um outro problema é o facto da natureza das listras projetadas serem muito pequenas em tamanho, sendo os *scanners* de luz estruturada apenas capazes de calcular a profundidade numa área limitada de digitalização [24].

Uma desvantagem, comum em todos os métodos óticos, consiste na dificuldade de digitalização de objetos com superfícies transparentes e refletivas. As superfícies transparentes têm de ser revestidas com um fino verniz opaco para poderem ser digitalizadas [18], [25].

Algumas aplicações típicas para *scanners* de luz estruturada incluem: o reconhecimento de gestos, a navegação de robôs, a deteção de pessoas e a digitalização de objetos estáticos sem grande nível de detalhe [24].

2.4.3 Time-of-flight (TOF)

Os *Time-of-flight (TOF) scanners* utilizam uma luz laser ou infravermelha para digitalizar um objeto ou um ambiente. Este dispositivo emite um pulso de luz para o objeto que se deseja digitalizar e, após atingir a superfície do objeto parte do pulso é refletido na direção do sensor. Para se descobrir a distância entre o sensor e o objeto é necessário que o sistema seja capaz de medir o tempo necessário para a luz percorrer uma viagem de ida e volta, ou seja, o tempo que o sinal leva para ir do transmissor ao objeto e retornar ao sistema. Como a luz viaja a velocidades extremamente elevadas, é requerido um circuito muito sofisticado para medir o tempo de retorno do feixe de luz. Portanto, para se obter uma digitalização completa de um objeto é necessário recorrer a espelhos rotativos para medir com precisão as diversas coordenadas *3D* de uma superfície [3], [13], [26], [27]. Um exemplo de uma configuração de um sistema *TOF* é apresentado na *Figura 8*.

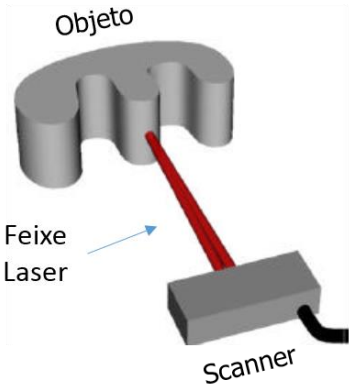


Figura 8: Representação de um sistema *Time-of-Flight* (*TOF*) [27]

Esta técnica é designada por *TOF* pelo simples facto de utilizarem o tempo como medida de distância [27]. O cálculo da distância a partir do tempo é baseado na bem conhecida fórmula:

$$v = \frac{d}{t} \tag{2.1}$$

$v: velocidade\ (m/s)$

$d: distancia\ (m)$

$t: tempo\ (s)$

A partir da velocidade da luz e do tempo decorrido entre a emissão do feixe de luz (laser ou infravermelha) e a receção da mesma luz após refletida na superfície do objeto, conseguimos descobrir a distância a que se encontra. A distância à superfície do objeto (*d*) é metade da distância percorrida pelo feixe de luz laser (*Figura 9*). Logo, esta equação pode ser rearranjada da seguinte forma:

$$d = \frac{(Tp * c\ )}{2} \tag{2.2}$$

$c: velocidade\ da\ luz\ (m/s)$

$Tp: tempo\ percorrido\ (s)$

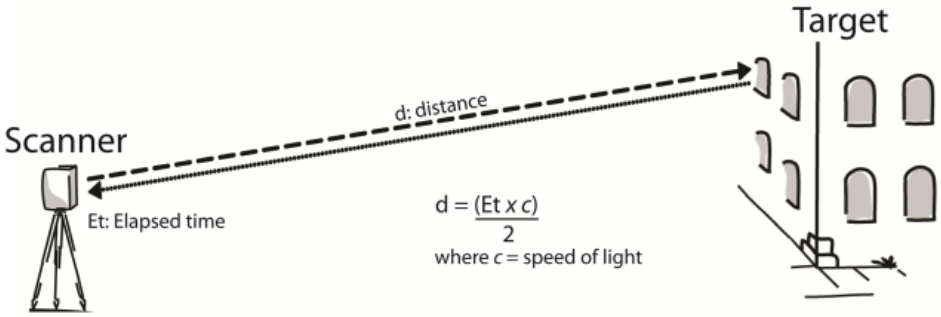


Figura 9: Método de cálculo da distância ao objeto [28]

Além dos valores de profundidade, os *TOF scanners* também fornecem os valores de intensidade, que representa a quantidade de luz enviada de volta por um ponto específico. O tempo decorrido entre a emissão e receção do feixe de luz pode ser medido por meio de modulação de onda contínua ou pulsada. Contudo, existem dispositivos *TOF* com base em ambas as técnicas.

Considerando os sistemas *TOF* baseados na modulação discreta de um pulso a profundidade pode ser medida diretamente pelo tempo que a luz demora a fazer a viagem de ida e volta. Neste caso, o tempo de chegada necessita de ser detetado de forma muito precisa. Para tal, convém que os pulsos de luz sejam curtos, mas com rápidos tempos de subida e de descida. Por outro lado, nos sistemas *TOF* baseados na modulação de uma onda contínua é imprescindível medir a diferença de fase entre o sinal enviado e o recebido. O sinal pode ter diversas formas, como por exemplo sinusoidal ou onda quadrada. A correlação cruzada entre os sinais enviados e recebidos permite estimar a fase, que está diretamente relacionada com a distância, se a frequência de modulação for conhecida [26], [29]

A relação entre o deslocamento de fase e o tempo percorrido é dado pela seguinte equação:

$$\phi = 2\pi f \tau \tag{2.3}$$

$\phi: deslocamente\ de\ fase\ (rad)$

$f: frequência\ de\ modulação\ (MHz)$

$t: tempo\ percorrido\ (s)$

Por sua vez, partindo da relação anterior e, através da *equação 2.2*, esta fórmula pode ser rearranjada da seguinte forma:

$$d = \frac{c}{2f} * \frac{\phi}{2\pi} \tag{2.4}$$

Na *Tabela 1* é feita uma breve comparação entre estes dois métodos de *TOF scanners*, salientando as vantagens e desvantagens de cada um.

Tabela 1: Vantagens e desvantagens modulação de onda pulsada e contínua [30]

TOF Scanners	**Vantagens**	**Desvantagens**
Modulação de onda pulsada (Pulse-based)	• Médio e longo alcance (2m – 1000m)	• Menos preciso • Aquisição de dados mais lento • Maior ruído
Modulação de onda contínua (Phase-shift)	• Mais preciso • Aquisição de dados mais rápido • Menor ruído • Mais leves	• Apenas médio alcance

A vantagem dos *TOF scanners* é o facto de estes serem capazes de operar em distâncias muito longas, na ordem dos quilómetros. Estes *scanners* são adequados para a digitalização de grandes estruturas, como por exemplo edifícios e ambientes geográficos detalhados. A grande desvantagem dos *TOF scanners* é a sua precisão. Devido à elevada velocidade da luz, é difícil cronometrar o tempo de ida e volta e a precisão das medições de distância é relativamente baixa, na ordem de milímetros. Para além disso, esta tecnologia sofre de outros problemas como baixa relação sinal-ruído, múltiplos reflexos de luz e dispersão do sinal. Tudo isto afeta os valores de profundidade registados e, portanto produz pontos *3D* errados, cuja confiabilidade depende de vários parâmetros da cena. Os *scanners* por triangulação a laser são exatamente o oposto. Estes sistemas têm uma distância de operação limitada, mas a sua precisão é relativamente elevada [10], [31]

2.5 Sem Contacto Passivo

Como foi dito anteriormente, este método de digitalização, como o próprio nome indica, não têm qualquer tipo de contacto físico com o objeto a ser digitalizado. Os *scanners* sem contacto passivos não emitem qualquer tipo de radiação por eles próprios, mas em vez disso baseiam-se na deteção de radiações ambiente refletidas. A maioria das soluções deste tipo detetam a luz visível porque é uma radiação ambiental facilmente disponível. Outros tipos de radiação, como por exemplo infravermelhos também poderiam ser utilizados. As soluções passivas podem ser muito baratas, porque na maioria dos casos não necessitam de hardware especializado, mas apensas de câmaras digitais simples. Algumas técnicas deste tipo incluem sistemas estereoscópicos, sistemas fotométricos, entre outros.

2.5.1 Sistemas estereoscópicos

Sistemas estereoscópicos são sistemas passivos que adquirem a forma tridimensional de um objeto utilizando apenas câmaras. Estes podem usar uma ou várias câmaras para aquisição do mesmo objeto, mas devem adquirir sempre pelo menos duas imagens do objeto a partir de diferentes pontos de vista. Este método baseia-se na visão estereoscópica humana. A posição e a informação de profundidade é obtida através da correspondência dos pixéis de ambas as imagens, e posteriormente a sua forma *3D* reconstruída por triangulação. Em alguns métodos, objetos de pouca textura tornam-se difíceis de analisar, porque poucos pontos característicos

aparecem na superfície do objeto [32]. Para ilustrar melhor este sistema, a *Figura 10* apresenta um esquema de um sistema estereoscópico simplificado.

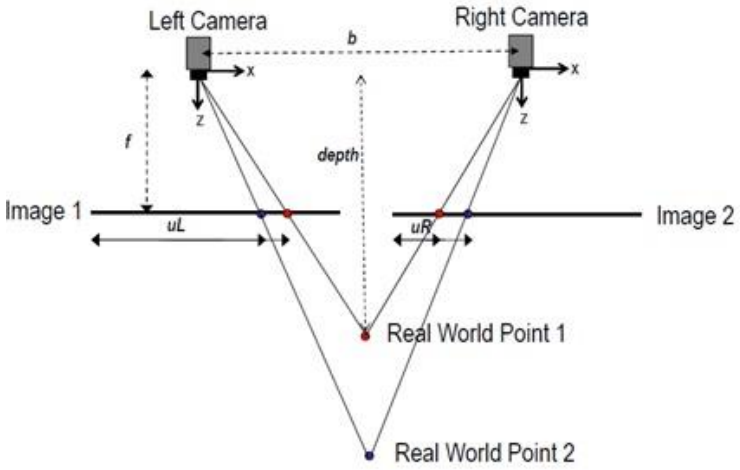


Figura 10: Representação de um sistema estereoscópico simplificado [24]

Ambas as câmaras estão montadas perfeitamente paralelas uma à outra, e possuem exatamente a mesma distância focal (*f*). A distância (*b*) corresponde à linha de base, ou seja, a distância entre as duas câmaras. O ponto *P1* representa um ponto no mundo real definido pela sua respetiva coordenada (*x, y, z*). U_L e U_R são as projeções do ponto *P1* do mundo real numa imagem adquirida pela câmara esquerda e direita, respetivamente. A distância entre estes dois pontos projetados é conhecida como disparidade. A partir da subtração das diferenças das duas as imagens é possível produzir um mapa de disparidade (*Figura 11*). O valor de disparidade pode ser usado para calcular as informações de profundidade, que é distância entre o ponto *P1* e o sistema [24], [33]. A profundidade dos pontos é determinada pela seguinte fórmula:

$$profundidade = fc * \frac{b}{disparidade} \tag{2.5}$$

$fc: distância\ focal$

$b: distância\ entre\ as\ duas\ câmaras$

Figura 11: Imagem proveniente da câmara esquerda e direita, respetivamente, e o resultante mapa de disparidade [24]

Os *scanners* estereoscópicos passivos são semelhantes aos *scanners* de luz estruturada, onde o projetor é substituído por uma câmara. Os sistemas estereoscópicos são mais apropriados para aplicações nas quais a instalação e localização das câmaras é fixa. As aplicações mais comuns para estes sistemas incluem navegação de veículos autônomos, robótica industrial, inspeção automatizada de peças, entre outras aplicações. A vantagem deste sistema em relação aos sistemas ativos reside no facto de não emitir qualquer tipo de radiação, sendo um método menos complexo a nível de hardware. Nesse sentido, a grande vantagem dos *scanners* sem contacto passivos é o custo. Esta tecnologia não necessita de quaisquer dispositivo ou máquina, só precisa de uma ou mais câmaras digitais. Esta técnica pode ser uma boa opção quando o orçamento para o dispositivo de digitalização *3D* for reduzido. A desvantagem destes *scanners* é a necessidade de objetos com diversas texturas e cores para o correto funcionamento desta técnica. Caso os objetos tenham poucas texturas e cores, ou seja poucos pontos característicos, é difícil obter uma correta combinação de pixéis, que pode se traduzir em inúmeros erros. Outro problema dos *scanners* estereoscópicos é a aquisição de amostras dispersas e com bastante ruído associado [33].

2.5.2 Sistemas fotométricos

Os sistemas fotométricos utilizam uma única câmara para obterem diversas imagens a partir de um ponto de vista fixo com diferentes condições de luminosidade. Em cada imagem, a cena é iluminada a partir de diferentes ângulos (*Figura 12*). Este método é altamente eficaz para separar a forma *3D* de um objeto a partir da sua textura a *2D*. Esta técnica é especialmente útil para a deteção de pequenos defeitos na superfície [34] [35].

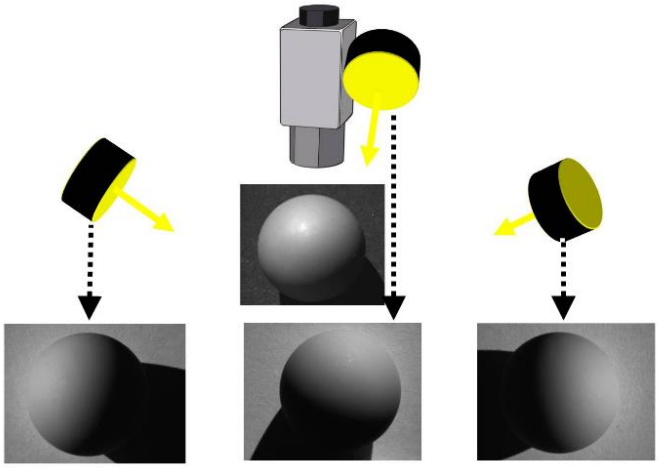


Figura 12: Sistema fotométrico: configuração da iluminação de cada imagem [34]

Esta técnica é utilizada para recuperar a forma de um objeto a partir de uma série de imagens tiradas em diferentes condições de iluminação. A forma da superfície do objeto é definida por um mapa de gradiente, o qual é constituído por uma matriz de normais da superfície (*surface normals*). Cada normal define a orientação de cada ponto da superfície. A *Figura 13* ilustra um diagrama com a normal de um único ponto da superfície de um objeto. A partir deste diagrama, pode também ser visto que a orientação da normal é definida por uma componente *x*, *y* e *z* [36].

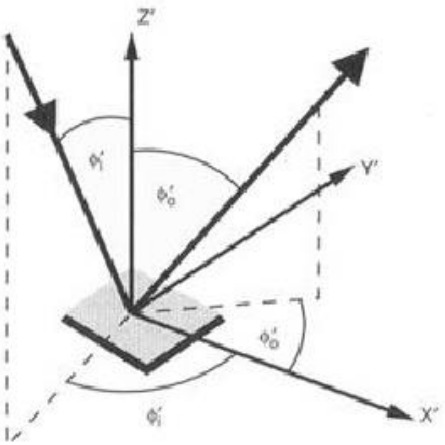


Figura 13: Esquema de obtenção da normal de um ponto da superfície [36]

Como o sistema fotométrico utiliza imagens de computador, cada normal é atribuída à unidade mais pequena de uma imagem, o píxel. Portanto, o número de normais depende do número de pixéis contidos na imagem. Para recuperar a normal de uma superfície foi definido

como sendo necessárias pelo menos 3 imagens, cada uma delas com uma condição de iluminação diferente. É possível usar vários pontos de vistas para capturar uma maior superfície de um objeto, no entanto o método mais simples e preciso utiliza um ponto de vista constante. Além do mapa de gradiente, este sistema também pode ser utilizado para produzir uma imagem de albedo. Esta imagem é de iluminação invariante e descreve a textura da superfície do objeto, sem forma [36].

Para produzir modelos *3D* a partir dos dados fotométricos é necessário integrar o mapa de gradiente para produzir a altura do mapa. A altura do mapa, também conhecida como profundidade, define a altura relativa para cada parte da superfície, sendo que os maiores valores representam as superfícies mais próximas da câmara no momento de captura. Para fazer esta integração existem duas técnicas que podem ser usadas para produzir um modelo *3D*, sendo estas técnicas locais ou globais. As técnicas de integração locais começam num local dentro dos dados de gradiente, frequentemente o centro, e vão progressivamente realizar a integração em valores vizinhos, até que uma integração completa seja alcançada. Por outro lado, as técnicas globais integram o mapa de gradiente como um todo, usando múltiplas verificações, a fim de reconstruir a totalidade da superfície. Para este tipo de sistemas, a integração global é geralmente empregue devido ao elevado ruído presente nas imagens de computador [36].

Os sistemas fotométricos são muito benéficos porque através de diferentes direções de luminosidade, a superfície do objeto pode ser capturada com grandes detalhes, como resultado disso, um perfeito e preciso objeto tridimensional pode estar presente. Contudo, esta técnica tens dois grandes problemas associados, que são as sombras e a luz ambiente. As sombras podem ocorrer quando as fontes de luz não são capazes de atingir toda a superfície do objeto, muitas vezes causadas por saliências da peça alvo. Não importa o quão cuidadoso é feito o arranjo das fontes de luz, as sombras são um fenômeno quase inevitável, especialmente em objetos com geometrias complexas. Isto pode resultar na perda de dados ou erro no cálculo das normais da superfície. A segunda questão baseia-se no facto de este sistema precisar de ser utilizado num ambiente controlado, ou seja, a quantidade de luz ambiente tem de ser mínima, caso contrário pode causar cálculos incorretos das normais de superfície [36], [37].

2.6 Sumário

Para proceder à digitalização de um determinado objeto utiliza-se um *scanner 3D*. Um *scanner 3D* é um dispositivo que analisa um objeto do mundo real para recolher dados sobre a sua forma e, possivelmente, a sua aparência. Estes dados após serem recolhidos podem então ser utilizados para construir modelos tridimensionais.

Existem uma série de tecnologias capazes de recolher a informação *3D* de objetos. Estas tecnologias podem ser divididas em dois tipos: contacto e não contacto. Os *scanners 3D* por contacto precisam de contacto físico para obter a informação *3D*. Eles geralmente têm alta precisão, mas necessitam de muito tempo para fazerem uma digitalização completa e podem danificar o objeto durante o processo. Os *scanners* sem contacto geralmente utilizam um laser ou luz estruturada para irradiar um objeto e obter a sua reflexão para calcular a informação *3D*. Comparando com os *scanners* por contacto, a digitalização é rápida, o seu funcionamento é simples e não danificam o objeto digitalizado. Contudo, a desvantagem é que estes sistemas não são tão precisos. Eles são considerados sistemas ativos se emitirem algum tipo de radiação, tais como, laser, caso contrário, se não emitirem qualquer tipo de radiação, mas detetarem as radiações ambiente refletidas são considerados sistemas passivos. No seguinte esquema as diversas técnicas são agrupadas nas respetivas tecnologias de digitalização *3D* (*Figura 14*).

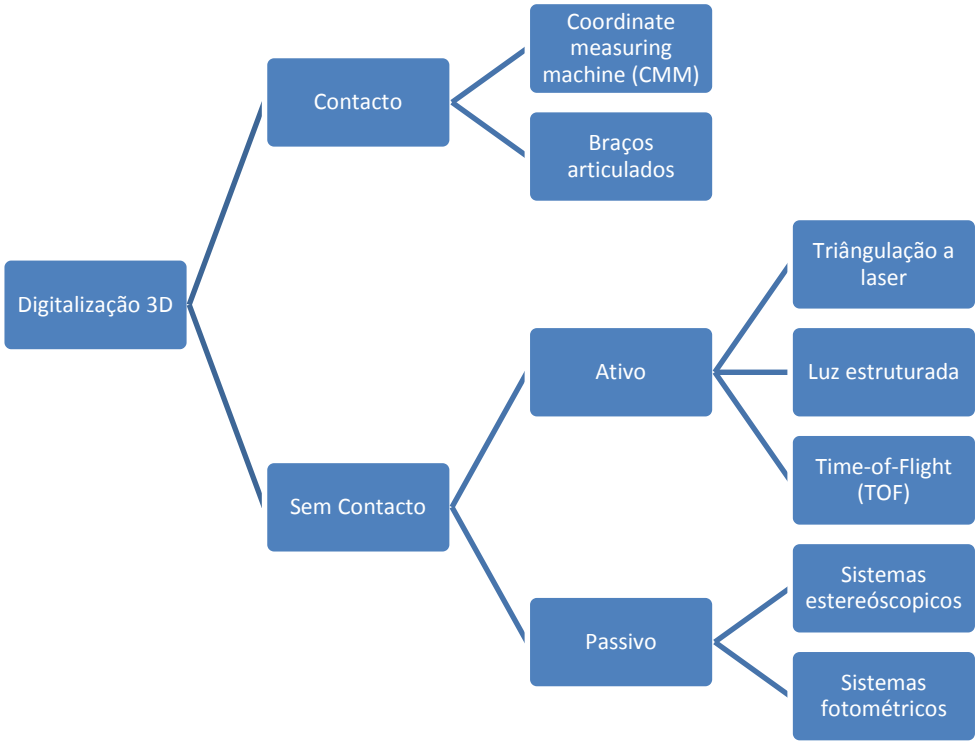


Figura 14: Caracterização das diversas tecnologias e técnicas de digitalização *3D*

Embora exista uma grande variedade de métodos que podem ser utilizados para realizar a digitalização *3D* de objetos, nenhum deles é perfeito, assim sendo cada método tem as suas limitações, vantagens e custo diferentes. Em seguida, as diversas técnicas são examinadas, sendo apresentado e discutido os pontos fortes e fracos da cada uma delas.

Basicamente, o maior problema dos *scanners 3D* por contacto é a necessidade de contacto durante o processo de digitalização, que às vezes pode danificar o objeto que se pretende digitalizar. Um outro problema é a baixa velocidade de digitalização, no entanto esta tecnologia garante um resultado muito preciso e exato, possivelmente a tecnologia que garante a informação tridimensional mais valiosa, sendo esta a chave do sucesso destes sistemas. Na *Tabela 2* é feita uma comparação detalhada entre os dois métodos de digitalização por contacto analisados neste documento, salientando as vantagens e desvantagens de cada um.

Tabela 2: Vantagens e desvantagens dos *scanners* de contacto

***Scanners 3D* por contacto**	**Vantagens**	**Desvantagens**
Coordinate measuring machine (CMM)	• Elevada precisão • A maioria opera automaticamente	• Processo lento • Possibilidade de danificar o objeto • Custo elevado • Sem portabilidade
Braços articulados	• Maior portabilidade • Preço mais acessível do que a CMM	• Operação manual • Processo lento • Menor precisão do que a CMM

A grande vantagem dos *scanners 3D* sem contacto é a possibilidade de obtenção de milhões de pontos ao longo de uma área ou região de um objeto de forma bastante rápida. Devido ao constante avanço da tecnologia de sensores e projetores, os *scanners* sem contacto estão a atingir precisões semelhantes aos seus rivais, os scanners por contacto.

As técnicas sem contacto ativas de digitalização tridimensional são o método mais eficaz para aquisição da forma *3D* de um objeto, permitindo alta taxa de amostragem e medições rápidas. No entanto, todos os métodos óticos de digitalização *3D* têm uma desvantagem que se baseia na reconstrução de uma superfície que contém oclusões independentes ou componentes desconetados devido a diferentes ângulos de incidência de sensores (câmaras) e/ou projetores, onde partes da superfície em questão permanece oculta.

Na *Tabela 3* é feita uma comparação detalhada entre os métodos de digitalização sem contacto ativo analisados neste documento, salientando as vantagens e desvantagens de cada um.

Tabela 3: Vantagens e desvantagens dos *scanners* sem contacto ativos

***Scanners 3D* sem contacto ativos**	**Vantagens**	**Desvantagens**
Triangulação a laser	• Boa precisão • Elevada resolução a curtas distâncias • Rapidez de digitalização • Maior portabilidade	• Grande dependência dos níveis de luminosidade (menor que sistemas de luz estruturada) • Problemas com a textura dos materiais • Prejudicial para os olhos e outras partes do corpo
Luz estruturada	• Processo rápido (maior do que sistemas a laser) • Adequado para objetos pequenos e estáticos • Elevada precisão (maior do que sistemas a laser)	• Limitada pela área de digitalização • Problemas com a textura dos materiais • Limitados pela intensidade luminosa • Reduzida portabilidade
Time-of-Flight (TOF)	• Distâncias médias e longas • Adequado para grandes objetos	• Baixa precisão • Maior ruido associado

O ponto forte dos *scanners 3D* sem contacto passivos é facto de não necessitarem de emitir qualquer tipo de radiação, em vez disso utilizam apenas hardware simples e barato para capturarem a imagem, frequentemente uma câmara digital. Esta tecnologia é adequada para projetos com orçamento reduzido. Contudo, assim como nos *scanners 3D* sem contacto ativos, são indispensáveis condições de luminosidade bastante controladas, caso contrário pode resultar na ausência de informação em algumas partes da superfície do objeto. Outro problema é a textura e características dos objetos. Muitas vezes, pode ocorrer uma falha ao identificar as correspondências das características das imagens, o que pode resultar em buracos na superfície, devido a ausência de informação. Na *Tabela 4* é feita uma comparação detalhada entre os dois métodos de digitalização sem contacto passivos analisados neste documento, salientando as vantagens e desvantagens de cada um.

Tabela 4: Vantagens e desvantagens dos *scanners* sem contacto passivos

***Scanners 3D* sem contacto passivos**	**Vantagens**	**Desvantagens**
Sistemas estereoscópicos	• Não emite qualquer tipo radiação • Sistema simples • Custo reduzido	• Problemas com a textura dos materiais (necessário elevada textura para uma boa digitalização) • Falhas de correspondência • Amostras dispersas e com ruído
Sistemas fotométricos	• Grande detalhe dos modelos • Sistema simples • Não emite qualquer tipo radiação	• Limitados pela intensidade luminosa (necessita de ambiente controlado) • Existência de sombras prejudiciais

Em resumo, cada tecnologia tem as suas próprias forças e fraquezas. A escolha da tecnologia mais adequada depende essencialmente do projeto para o qual é direcionado. Esta tem de ser capaz de responder em grande parte aos requisitos do mesmo, tais como os tipos de objetos que se pretende digitalizar, o ambiente de digitalização, a precisão, a resolução, a rapidez do processo, e ainda a aplicação a que a informação tridimensional se destina. Um fator preponderante para essa escolha é o custo do equipamento. De salientar que a razão entre o custo dos componentes e a qualidade do equipamento está sempre em causa.

Os dois métodos principais para obter os pontos tridimensionais da superfície de objetos são baseados em sistemas estereoscópicos e sistemas de medição de distância a laser. Contudo, a maioria dos instrumentos de medição de distâncias tridimensionais custam centenas e até milhares de euros, como por exemplo os lasers, as câmaras de profundidade, os projetores, entre outros. Embora consigam gerar mapas de profundidade de espaço de forma muito rápido e eficaz, em algumas dessas tecnologias o tratamento da informação *3D* é muito complicado para os utilizadores não profissionais. Logo, a utilização de algumas dessas tecnologias na reconstrução *3D* é limitada.

Nos últimos anos, os sistemas de visão artificial foram submetidos a uma substancial evolução tecnológica. A nível industrial, há um aumento de soluções propostas, devido a novas técnicas e algoritmos evoluídos. A maior parte destas aplicações estão baseadas em tecnologias de aquisição e processamento de imagem *3D*.

2.7 Microsoft Kinect

Em Novembro de 2010, a empresa *Microsoft* lançou no mercado o sensor de movimento *Kinect* desenvolvida para a *Xbox 360* e *Xbox One*. Esta tecnologia foi criada com o objetivo de permitir maior interação entre os jogadores e os jogos eletrónicos sem a necessidade de um controlador, através de uma interface natural usando gestos e comandos de voz. No entanto, o impacto do sensor *Kinect* se estendeu muito além da indústria de jogos. Com sua ampla disponibilidade e baixo custo, muitos investigadores e profissionais em ciência da computação, engenharia eletrónica e robótica estão aproveitando a tecnologia de reconhecimento e deteção para desenvolver novas formas de interagir com as máquinas e para executar outras tarefas, tais como ajudar as crianças com autismo e a apoiar médicos em salas de cirurgia (*Figura 15-a*). Para além da sua vasta aplicabilidade, *o Microsoft Kinect* é uma ferramenta eficaz para a aquisição de modelos tridimensionais. Devido ao seu hardware simples e económico tem vindo a emergir no mercado como uma solução de mapeamento e digitalização *3D* [38]–[40].

(a)

(b)

Figura 15: Exemplo de aplicações do *Microsoft Kinect*: análise de exames de pacientes durante a cirurgia [41] (a) e reconstrução 3D de uma pessoa [42] (b)

O *Microsoft Kinect* é constituído por um emissor de laser infravermelho, uma câmara de infravermelhos e uma câmara *RGB* (*Figura 16*). A câmara de infravermelhos e o emissor laser, utilizado como projetor de padrão infravermelho, representam um par estereoscópico para triangular pontos no espaço *3D*. A câmara *RGB* pode ser utilizada para texturizar os pontos *3D* ou reconhecer o conteúdo da imagem. Este equipamento possui ainda um conjunto de quatro microfones e um motor de inclinação que proporcionam a captura de movimentos *3D* de corpo inteiro, o reconhecimento facial, e recursos de reconhecimento de voz. Como resultado de um dispositivo de medição, o *Kinect* disponibiliza três imagens: a imagem infravermelha, a imagem *RGB*, e a imagem de profundidade invertida [43], [44].

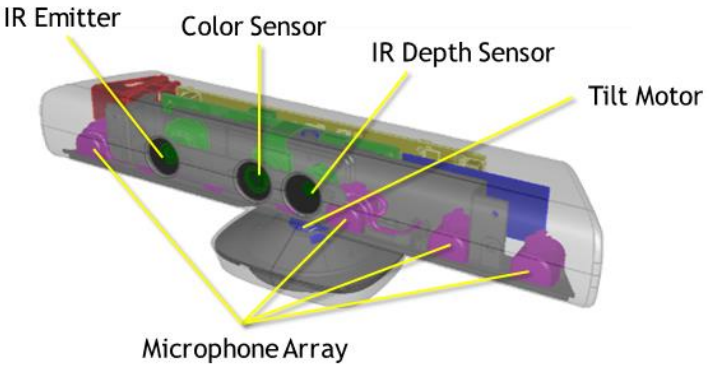


Figura 16: Sensores e outros componentes do *Microsoft Kinect* (vista interior) [45]

O emissor de laser emite um único feixe, que é dividido em múltiplos feixes por uma rede de difração para criar um padrão constante de manchas projetadas na cena (*Figura 18-a*). Este padrão é capturado pela câmara infravermelha e é correlacionado com um padrão de referência (*Figura 17*). O padrão de referência é obtido através da captura de um plano a uma distância conhecida do sensor (Z_0), e armazenada na memória do sensor. Quando o padrão é projetado sobre um objeto, cuja distância ao sensor pode ser menor ou maior do que a do plano de referência (Z_k), a posição da padrão na imagem de infravermelhos (o) será deslocada na direção da linha de base entre o projetor de laser e a perspetiva do centro da câmara infravermelha. Esses deslocamentos são medidos para todas as manchas por um processo de correlação de imagens simples, que produz uma imagem de disparidade. Para cada píxel a distância do sensor pode ser obtida a partir da disparidade correspondente (d) [44].

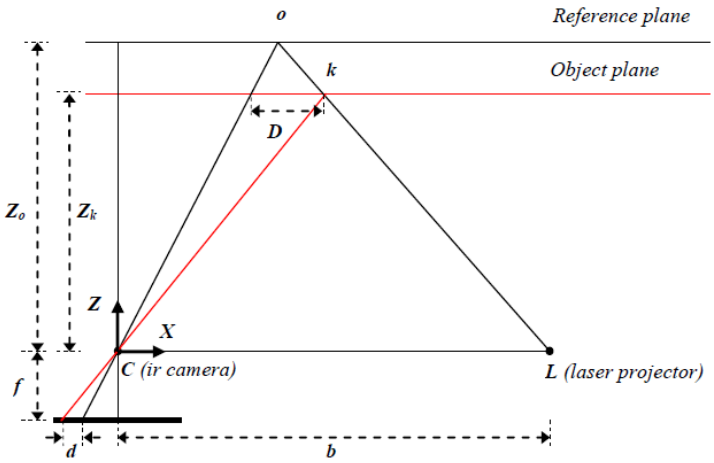


Figura 17: Relação entre a profundidade e disparidade medida [44]

A *Figura 18-b* ilustra a medição de profundidade a partir do padrão de manchas projetado sobre uma cena.

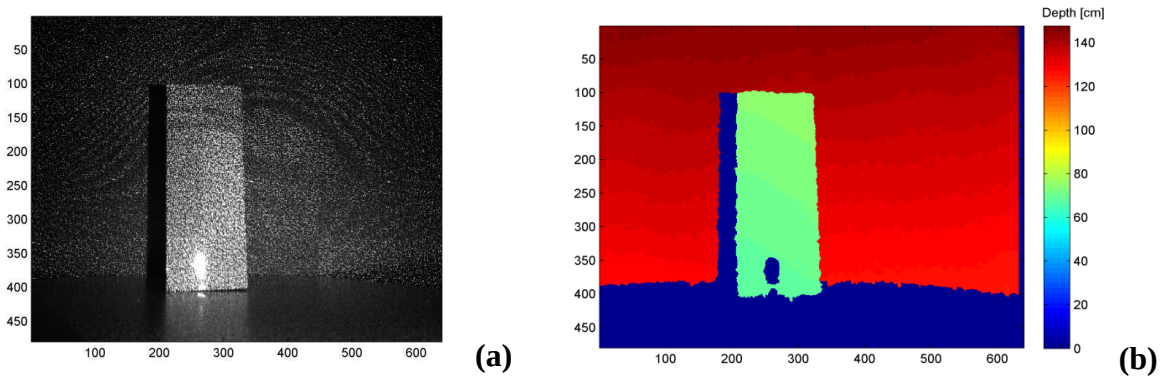


Figura 18: Imagem infravermelha do padrão projetado numa cena (a) Imagem de profundidade resultante (b) [44]

Na *Tabela 5* é descrita de forma resumida as principais vantagens e desvantagens *do Microsoft Kinect.*

Tabela 5: Vantagens e desvantagens do *Microsoft Kinect* [40]

Microsoft Kinect	
Vantagens	**Desvantagens**
• Informação de profundidade precisa; • Informação *3D* rápida; • *FPS* com 30Hz; • Faixa de operação razoável; • Baixo Custo; • Portátil.	• Sensível às condições de iluminação. Não é adequado para ambientes *outdoor*; • Mapa de profundidade não é corretamente detetado para distâncias inferiores a 0,7 m; • Não é possível detetar objetos transparentes ou altamente refletivos.

2.8 Exemplos de Scanners 3D

De seguida, são apresentados alguns exemplos de *scanners 3D* existentes que utilizam as diversas técnicas descritas neste documento. É feita uma breve análise das características de cada um dos sistemas, assim como as possíveis aplicações em que podem ser aplicados. Por fim, é descrito de mais forma detalhada alguns dos *sotfwares* que utilizam a tecnologia aplicada neste projeto de dissertação.

2.8.1 CRYSTA-Apex S

A *CRYSTA-Apex S* é uma *CMM* de alto desempenho desenvolvida pela empresa *Mitutoyo*. Especialmente desenhada e construída para garantir os requisitos europeus, o sistema controlado por *CNC* é um modelo versátil e fácil de utilizar. É uma máquina de elevada precisão cerca de 1,7 μm e vem equipado com um sistema de compensação de temperatura que garante precisão de medição sob condições de temperatura de 16 a 26 *ºC*. A velocidade máxima de medição é 8 mm/s e está preparada para medições com vários tipos de sondas. É um dispositivo frequentemente utilizado em processos de produção e montagem de peças [46]–[48].

Figura 19: *CRYSTA-Apex S* desenvolvido pela empresa *Mitutoyo [46]*

2.8.2 FARO Gage Plus

O *FARO Gage Plus* é um sistema portátil de digitalização tridimensional. Foi criado com o objetivo de acabar com a dependência das *CMM's* fixas e caras. Devido à sua portabilidade este sistema pode ser utilizado em qualquer lugar, sendo configurado em segundos, permitindo a medição de peças de forma fácil, rápida e precisa. Este equipamento garante uma precisão até 0,005 mm numa área de trabalho não superior a 1,2 m e possibilita ainda a utilização de diversas sondas. Pode ser utilizada nas mais diversas áreas, tais como a indústria metalúrgica, automóvel, aeroespacial, essencialmente para a análise dimensional, a inspeção e o alinhamento de peças [49], [50].

Figura 20: *FARO Gage Plus* [50]

2.8.3 MakerBot Digitizer

O *scanner 3D* desenvolvido pela *MakerBot* é excelente tecnologia, tendo sido elogiada pela *Popular Mechanics* como um dos principais produtos inovadores de 2013 [51]. Este equipamento foi projetado para uma fácil e rápida digitalização de objetos 3D. Este dispositivo é composto por dois lasers de linha, uma câmara *CMOS* com 1.3 Megapíxel e um motor de passo com um ângulo de passo de *0,9º*. A velocidade de digitalização é de aproximadamente 12 min. As outras características relevantes deste produto são a resolução de 0,5 mm e a precisão de 2,0 mm. Este sistema é limitado a peças com 20 cm de diâmetro e 20 cm de altura. Este *scanner* não é o mais apropriado para objetos em movimento nem para objetos com propriedades transparentes, brilhantes e/ou refletivas. Atualmente, o preço deste equipamento é de cerca de 700 euros [52], [53].

Figura 21: *MakerBot Digitizer* [52]

2.8.4 FACESCAN STT 3D

O *FACESCAN* é um *scanner 3D* com base em tecnologia de luz estruturada. Este sistema de luz estruturada é constituída por 2 câmaras digitais e um projetor. Embora tenha sido desenvolvido para digitalizar faces, pode ser utilizado para produzir modelos tridimensionais de objetos e superfícies complexas. Este sistema é capaz de originar dados tridimensionais com uma precisão de 0.2 mm e uma resolução de 0.04 mm por píxel. O único requisito que este dispositivo requer é uma condição de iluminação uniforme de espaço de trabalho [54].

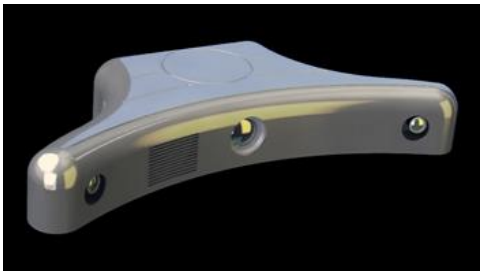


Figura 22: *FACESCAN 3D* desenvolvido pela *STT Engineering & Systems* [54]

2.8.5 DAVID Structured Light Scanner SLS-2

O *David Scanner SLS-2* é um outro sistema de digitalização *3D* baseado na tecnologia de luz estruturada, mas ao contrário do *FACESCAN* este apenas utiliza uma câmara. A dimensão dos objetos que se pretendem digitalizar é limitada (10 mm – 500 mm), sendo a precisão deste sistema aproximadamente 0.1% do tamanho do objeto. A velocidade da digitalização é caraterizada por ser rápido, entre 2 e 4 s. Os resultados podem ser exportados em formatos de arquivo *3D* comum (*OBJ, STL, PLY*) e processados em outras aplicações, tais como, prototipagem rápida, apresentação de produtos, arqueologia e património cultural, animações de computados, entre outras. O custo desta solução é de 2350 euros [55]–[57].

Figura 23: *DAVID Structured Light Scanner SLS-2*

2.8.6 Leica ScanStation C10

O *Leica Scan Station C10* é um *TOF scanner* que inclui uma elevada precisão para digitalizações de longo alcance e com a vantagem associada de ser um processo rápido. Este equipamento utiliza um laser se cor verde com modulação pulsada. As especificações deste sistema indicam que a precisão para uma única medida é de 6 mm na posição e 4 mm de profundidade (para distâncias até 50 m). A chave deste equipamento está no novo *design* do espelho que automaticamente gira ou oscila para otimizar a produtividade. Com um alcance até 300 m, este sistema é rentável para aplicações topográficas, monitoramento de pesquisas, sendo ainda ideal inúmeras aplicações de arquitetura, engenharia, planeamento e investigação. O custo deste equipamento é de 35 mil euros [58], [59].

Figura 24: *Leica ScanStation C10* [58]

2.8.7 Fuel3D

O *Fuel3D* é o primeiro *scanner* do mundo a combinar câmaras estereoscópicas com imagens fotométricas para capturar e processar arquivos em segundos. É um equipamento portátil, sendo denominado como um sistema de imagem *3D point-and-shot* que consegue capturar a uma elevada resolução a informação *3D* e a cor de objetos. A resolução do *Fuel3D* varia com a distância do sistema ao objeto, sendo a máxima alcançada de aproximadamente 250 μm por amostra. Este *scanner* é apropriado para objetos com bastantes texturas, e funciona bem com outros tipos de objetos, tais como artefactos, pedras, tecidos, rostos e outras partes do corpo, entre outros. Este equipamento está disponível por um preço de 950 euros [60], [61].

Figura 25: *Fuel3D* [60]

2.8.8 The Structure Sensor

A empresa *Occipital* desenvolveu o *Structure Sensor*, o primeiro sensor *3D* do mundo para dispositivos móveis. Entretanto, a empresa *Google* está a desenvolver o *Projeto Tango* [62] que inclui *tablets* e telefones capazes de criar mapas *3D* do seu ambiente em tempo real. Este projeto, no entanto, não parece perto de comercialização.

Com o *Structure Sensor* conectado a um dispositivo móvel, é possível caminhar em redor do mundo e imediatamente capturar modelos *3D* de objetos e pessoas para importação em *CAD* e para a impressão *3D*. Este sensor tem um alcance otimizado para telemóvel que pode ir de 40 cm até mais de 3,5 m. A precisão deste equipamento é tipicamente 1% da distância medida. Possui uma fonte de alimentação própria capaz de fornecer até 4 horas de uso ativo e 1000 horas de standby. Ao contrário dos projetores de luz estruturada infravermelhos, os *LED's* infravermelhos emitem uma luz uniforme que permite capturar o mundo em infravermelho. Por último, resta salientar que o preço deste equipamento é de cerca de 300 euros [63], [64].

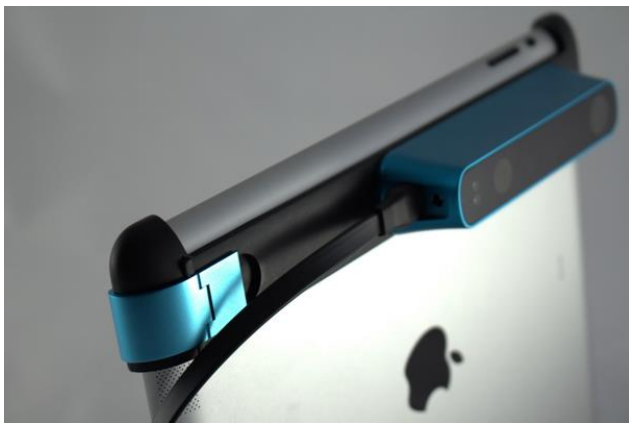
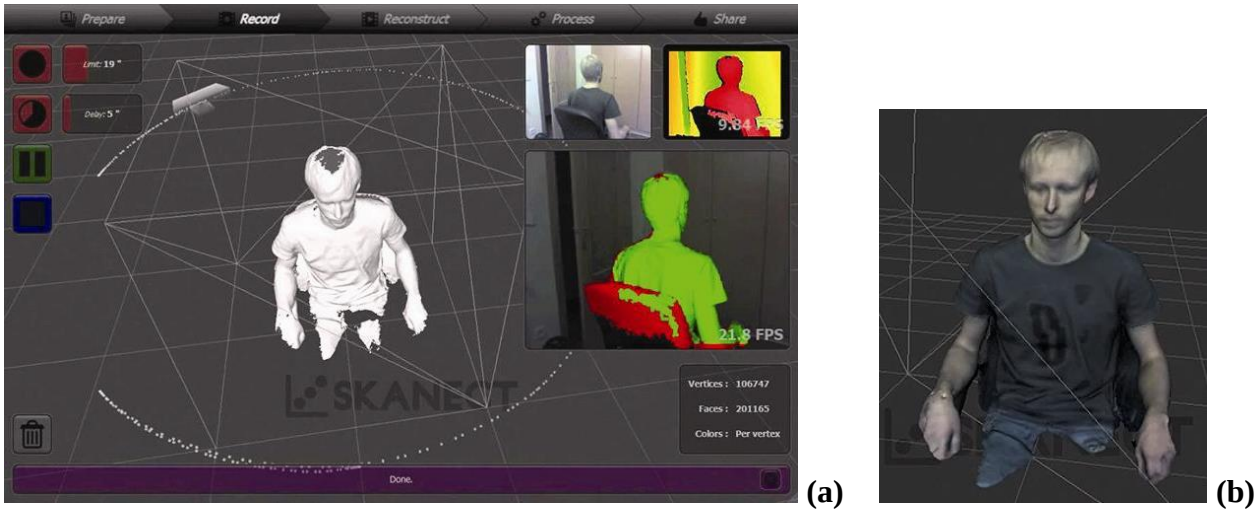


Figura 26: *The Structure Sensor* torna o *iPad* num *scanner 3D* [65]

2.8.9 Skanect

O *Skanect* transforma o sensor *Microsoft Kinect* ou a *Asus Xtion* num *scanner 3D* de custo reduzido capaz de criar modelos *3D* de cenas em poucos minutos. Este software permite a digitalização completa ou parcial de uma cena, com a vantagem de possuir feedback em tempo real em baixa, média e alta qualidade. A nível de processamento tem algumas características importantes, tais como a simplificação do modelo, a remoção de pequenas partes indesejáveis, o preenchimento de buracos, o corte de planos para otimizar a impressão *3D*, entre outras. Os resultados podem ser exportados para softwares e impressoras *3D* nos diversos formatos (*.stl, .obj, .ply, .vrml*). Este sistema é apropriado para digitalizações de partes do corpo, objetos e espaços. A versão *Pro* desta solução custa 100 euros [66].



(a) **(b)**

Figura 27: Skanect - Software [67] (a) Exemplo de modelo *3D* resultante [68] (b)

2.8.10 Artec Studio 9

O *Artec Studio 9* é um software de digitalização que não só trabalha com *scanners 3D* profissionais como o *Artec Eva*, mas também permite a digitalização *3D* de baixo custo a partir de sistemas como o *Kinect* para *Xbox/Microsoft*, o *Asus Xition/Xition Pro* ou outros sensores baseados em tecnologia *Primsense*. O software é projetado para ser intuitivo, com uma gama completa de ferramentas que permitem a digitalização e pós processamento dos dados digitalizados de forma rápida e fácil [69], [70].

Figura 28: *Artec Studio 9* com *Microsoft Kinect* para digitalização *3D* [71]

Este sistema oferece duas diferentes formas de digitalização. Um delas baseia-se na fusão em tempo real, ou seja, o software compara as diferentes capturas *3D* para encontrar as mesmas características geométricas e une estas capturas enquanto a digitalização é feita. A outra maneira caracteriza-se pela procura de texturas idênticas. O software analisa as características textuais e geométricas que são utilizadas para posicionar as capturas juntas, em tempo real, e só depois é que a fusão é feita. De modo a garantir a melhor precisão possível, o *Artec Studio* 9 caracteriza-se por integrar algoritmos avançados de otimização [69], [70].

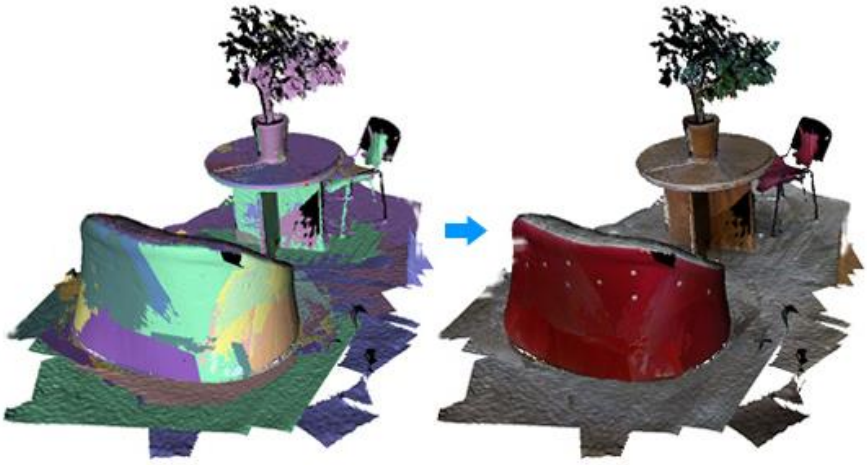


Figura 29: *Artec Studio 9* – Resultados texturizados e otimizados [70]

Um vasto leque de ferramentas são ainda incorporadas neste software, como por exemplo diversas métodos de edição e algoritmos de processamento avançados (filtros, preenchimento de buracos, suavização dos contornos e dos modelos, entro outros). Para além disso, o *Artec Studio 9* contem ferramentas de medição e análise de modelos *3D*. Este sistema de digitalização *3D* tá avaliado em 500 euros [69], [70].

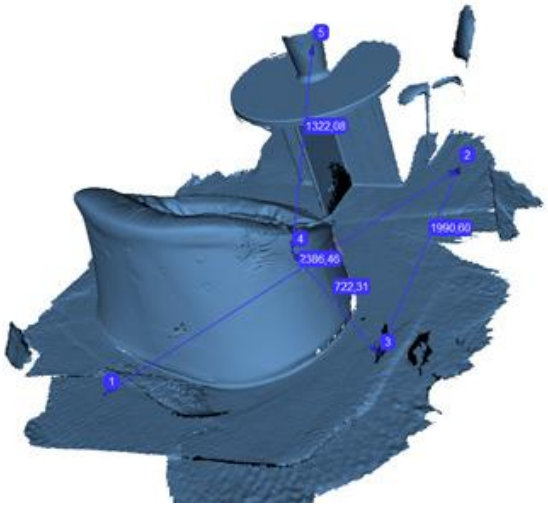


Figura 30: *Artec Studio 9* – Ferramenta de medição [70]

3. FUNDAMENTOS TEÓRICOS

No presente capítulo é apresentada a fundamentação teórica da metodologia adotada neste projeto de dissertação. A escolha da tecnologia, do software e das bibliotecas utilizadas neste projeto de investigação, assim como os métodos e algoritmos implementados com vista à resolução do problema proposto, serão descritos e fundamentados nesta seção.

3.1 Microsoft Kinect

No capítulo anterior, o sensor *Microsoft Kinect* é descrito como uma simples e económica ferramenta para aquisição de modelos tridimensionais. Nesta secção, é apresentada de forma mais pormenorizada as especificações deste sensor e, deste modo é possível descobrir os benefícios e limitações deste dispositivo na extração da informação *3D* de uma cena.

3.1.1 Especificações

Como mencionado anteriormente, o *Microsoft Kinect* é constituído por um sensor de profundidade, desenvolvido com a tecnologia da empresa *PrimeSense*, com base no método de luz estruturada infravermelho. Este dispositivo é ainda composto por uma câmara *RGB*, um motor de inclinação, um acelerómetro e por um conjunto de microfones. No entanto, o micro processador da *PrimeSense* é o núcleo de processamento do *Microsoft Kinect* (*Figura 31*) [72].

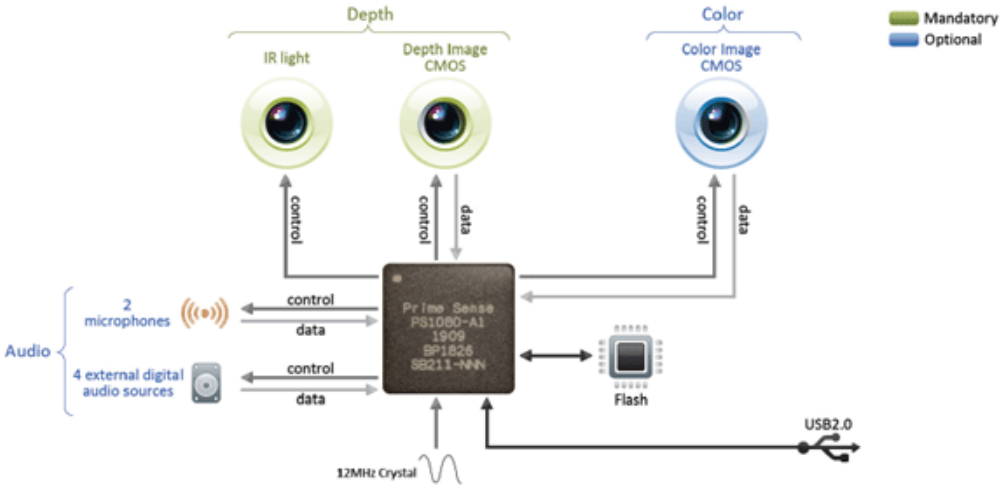


Figura 31: Diagrama dos componentes da tecnologia *PrimeSense* [73]

Na *Tabela 6* é descrita as principais características dos diversos constituintes que incorporam o sensor *Kinect*.

Tabela 6: Principais características do sensor *Microsoft Kinect* [72]

Principais características	Xbox 360 Kinect Sensor
Resolução câmara RGB	*640 x 480 píxeis*
Resolução câmara de profundidade	*640 x 480 píxeis*
Gama de operação	*0.8 m – 3.5 m*
Campo de visão	*58º horizontal / 45º vertical*
Ângulo de inclinação vertical	*± 27°*
Taxa de aquisição	*30 frames por segundo (FPS)*
Entradas áudio	*4 (16 kHz)*
Conexão	*USB 2.0*

3.2 OpenNI/ PrimeSense Drivers

O *OpenNI* (*Open Natural Interaction*) é um *framework open-source* multi-plataforma desenvolvido no sentido de permitir uma natural interação com um dispositivo ou uma tecnologia através de simples gestos ou comandos de voz. Este *framework* foi criado em Novembro de 2010 por um conjunto de organizações, sendo a *PrimeSense* um dos maiores impulsionadores deste projeto. Após o desenvolvimento do *OpenNI*, a *PrimeSense* lançou os seus próprios drivers *open-souce* e ainda o *middleware NITE*, responsável pelo rastreamento de movimento (*motion tracking*) [72], [74].

Em suma, este *framework* foi desenvolvida no sentido de controlar as interações naturais tridimensionais de sensores (sensores de profundidade), como o *Microsoft Kinect* ou o *ASUS Xtion*. No âmbito deste projeto de dissertação, o *OpenNI* e os drivers desenvolvidos pela *PrimeSense* são utilizados para obter os dados a partir do dispositivo *Microsoft Kinect*, que posteriormente serão processados recorrendo à biblioteca *PCL (Point Cloud Library)* [74].

(a) **(b)**

Figura 32: Logótipo do *OpenNI* [75] (a) Logótipo da *PrimeSense* [76] (b)

3.3 Point Cloud Library (PCL)

A *Point Cloud Library* é uma biblioteca *open-source* que contém um conjunto de algoritmos para o processamento de nuvens de pontos de diversas dimensões e geometrias tridimensionais. O objetivo desta ferramenta é fornecer suporte para todos os blocos de construção que uma aplicação *3D* necessita, a partir dos dados das nuvens de pontos obtidos por dispositivos de perceção tridimensional. É uma biblioteca multi-plataforma (*Windows*, *Linux*, *MacOS* e *Android/iOS*), sendo ainda capaz de ser integrada no *ROS (Robot Operating System*). A *PCL* é licenciada sob os termos da *Berkley Software Distribution (BSD)* [77].

Figura 33: Logótipo da *Point Cloud Library (PCL)* [77]

3.3.1 Estrutura da biblioteca

Para simplificar o seu desenvolvimento, a *PCL* está dividida em pequenos módulos ou bibliotecas que disponibilizam algoritmos e ferramentas para áreas específicas de processamento *3D*, que podem ser compiladas separadamente. Esta modularidade é importante para a distribuição da *PCL* em plataformas com recursos computacionais reduzidos ou com limitações de tamanho. Estas pequenas bibliotecas podem então ser combinadas para resolver eficazmente diversos problemas, tais como reconhecimento de objetos, registos de nuvens de pontos, segmentação e reconstrução da superfície [77]. A *Figura 34* representa um diagrama relacional das diversas bibliotecas que pertencem à *PCL*.

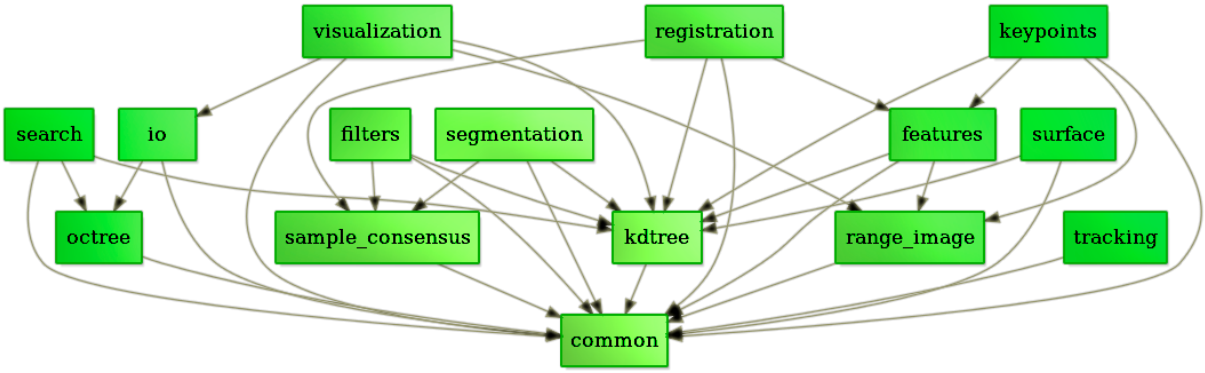


Figura 34: Diagrama relacional dos diversos módulos da *Point Cloud Library* [77]

3.3.2 Organização da biblioteca

Tal como mencionado na secção anterior, para conferir uma melhor organização a biblioteca *PCL* está divida em diversos módulos. De seguida, são descritos de forma geral alguns desses módulos (contidos na versão *1.8.0* desta biblioteca):

- **Módulo *common***: biblioteca base da *PCL*, que contém as estruturas de dados comuns e métodos utilizados na maior parte das bibliotecas *PCL*. O núcleo das estruturas de dados incluem a classe *Point Cloud* e uma diversidade de tipos de pontos que são usados para representar pontos, normais da superfície, valores de cor *RGB* e descritores de características (*features*). Contém ainda uma grande variedade de funções que permitem a computação da distância e norma, média e covariância, conversões angulares e transformações geométricas [78];

- **Módulo *features***: contém as estruturas de dados e os mecanismos para estimar as características *3D* (*features*) de uma nuvem de pontos (*point cloud*). Os *features* são representações de um determinado ponto ou posição tridimensional no espaço, sendo capazes de descrever os padrões geométricos com base nas informações disponíveis em torno do ponto. O espaço de dados selecionado na vizinhança de um ponto é normalmente referido como *k-neighborhood* [79]*;*

- **Módulo *filters***: contém mecanismos de remoção de *outliers* e de ruído para um conjunto de dados tridimensionais, assim como filtros genéricos para extração de subconjuntos de pontos ou para exclusão de partes do mesmo. Para além disso, este módulo proporciona também uma classe de *voxel-grid*, utilizada para fazer a diminuição de uma nuvem de pontos *(down-sample)* [80]*;*

- **Módulo *io*:** contém classes e funções para leitura e escrita de dados (ficheiros do tipo *.pcd, .ply, .vtk e .obj*), bem como a captura de nuvens de pontos a partir de uma variedade de dispositivos de deteção 3D (compatíveis com o *OpenNI*) [81];

- **Módulo *kd-tree:*** fornece uma estrutura de dados *kd-tree*, usando a implementação *FLANN*, que permite rápidas pesquisas entre os vizinhos mais próximos. A *Kd-tree* (árvore *k*-dimensional) é uma estrutura de dados de particionamento de espaço que armazena um conjunto de pontos *k*-dimensional numa estrutura de árvore [82];

- **Módulo *keypoints*:** implementa diferentes métodos de extração de pontos de interesse. Os *keypoints* (também referidos como pontos de interesse) são pontos numa imagem ou nuvem de pontos que são estáveis, distintos, e podem ser identificados através de um critério de deteção bem definido. Os métodos *Harris, Narf, SIFT e Uniform Sampling* são alguns dos algoritmos que estão implementados nesta versão [83];

- **Módulo *registration*:** inclui diversos algoritmos de registo de nuvens de pontos para conjuntos de dados organizados e desorganizados, incluindo *Iteractive Closest Point* (*ICP)*, métodos de procura e de rejeição de correspondências, métodos de estimação de transformações, entre outros [84];

- **Módulo *sample_consensus*:** é baseado em métodos de *SAmple Consensus (SAC), como RANSAC* e outros derivados. Estes métodos podem ser combinados de forma a detetar estruturas específicas, tais como esferas, cilindros, planos ou formas arbitrárias em nuvens de pontos desorganizadas [85];

- **Módulo *segmentation*:** contém algoritmos para segmentação de nuvens de pontos em distintos *clusters*. Estes algoritmos são os mais adequados para o processamento de uma nuvem de pontos que é composta por um número de regiões espacialmente isoladas. Em tais casos, a extração dos *clusters* é muitas vezes utilizada para separar a nuvem de pontos nos diversos constituintes, que podem ser processados independentemente [86];

- **Módulo *surface*:** contém métodos de reconstrução de superfície, *meshing, convex hulls, Moving Least Squares (MLS)*, entre outros [87];

- **Módulo *visualization*:** esta biblioteca permite a rápida visualização dos resultados dos algoritmos que operam sobre dados tridimensionais e nuvens de pontos. Baseada no *VTK*, possibilita a visualização de nuvens de pontos, normais, imagens de profundidade, correspondências, histogramas, entre outros [88], [89].

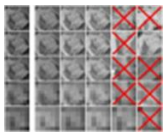
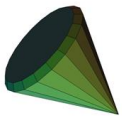
3.3.3 Dependências

A biblioteca *Point Cloud Library* depende ainda de uma série de outras bibliotecas como o *Boost*, o *Eigen*, o *FLANN* (*Fast Library for Approximate Nearest Neighbor*), o *VTK* (*Visualization ToolKit*). Estas bibliotecas são obrigatórias para a compilação e utilização da biblioteca *PCL*. Por outro lado, existem ainda outras dependências opcionais, tais como o *CUDA, o OpenNI* e o *QHull*, que não são obrigatórias, no entanto desativam um conjunto de funcionalidades dentro de alguns dos módulos da *PCL* mas compilam o resto da biblioteca que não requer a dependência [90].

O *Boost* [91] é uma biblioteca que está relacionada com a passagem de dados através de apontadores partilhados e com a utilização de *threads.* O *Eigen* [92] é uma biblioteca *C++* para operações relacionadas com álgebra linear: matrizes, vetores, funções numéricas e outros algoritmos relacionados. A biblioteca *FLANN* [93] é utilizada para a procura rápida dos vizinhos mais próximos em espaços de dimensão elevada. O software *VTK* [94] é um sistema *open-source* para computação gráfica 3D, processamento de imagem e visualização.

O *CUDA* [95] é uma biblioteca que permite a utilização da placa de vídeo do computador para efetuar diversos tipos de cálculos computacionais. Através do *framework OpenNI* é possível adquirir e interpretar os dados obtidos pelos dispositivos físicos, como o sensor *Microsoft Kinect*. O *QHull* [96] é uma biblioteca que contém algoritmos aplicados ao cálculo de estruturas relacionados com a reconstrução da superfície de um conjunto de pontos.

Tabela 7: Dependências obrigatórias e opcionais da biblioteca *PCL* [90]

Obrigatórias				
	Boost	***Eigen***	***FLANN***	***VTK***
Opcionais				
	CUDA	***OpenNI***	***QHull***	

3.4 Nuvem de pontos

Uma nuvem de pontos (vulgarmente denominada *point cloud*) é uma estrutura de dados que contém dados multidimensionais. Normalmente, as nuvens de pontos são utilizadas para representar os dados em três dimensões e, geralmente definem as coordenadas *x, y, z*. A aplicação mais comum para este conjunto de dados é a representação da superfície exterior de um objeto. Para além das coordenadas geométricas, as nuvens de pontos podem conter a informação da cor para cada ponto, das normais ou ambos. Quando alguma informação adicional é acrescentada à nuvem de pontos, a dimensão da nuvem aumenta. A *Figura 35* representa uma nuvem de pontos com informação *RGB* de um monumento histórico [97].

Figura 35: Nuvem de pontos com informação de cor *RGB* [98]

3.4.1 Tipos de Pontos

A biblioteca *PCL* tem uma variedade de tipos de pontos pré-definidos, que vão desde estruturas simples que apenas armazenam os dados correspondentes ao sistema de coordenadas tridimensional (coordenadas *x, y, z*) até representações *n*-dimensionais mais complexas como por exemplo a informação proveniente de descritores de características *3D* (*features*). Embora, os diversos tipos de pontos pré-definidos sejam capazes de suportar todos os algoritmos e métodos implementados na biblioteca, existe ainda a possibilidade do utilizador definir novos tipos consoante a sua necessidade. De seguida, são descritos alguns dos tipos mais utilizados para o processamento de nuvens de pontos:

- **PointXYZ** (Membros: *float x, y, z*): é um dos tipos de dados mais utilizados, representa apenas informações tridimensional *xyz*;

- **PointXYZI** (Membros: *float x, y, z, intensity):* representa a informação tridimensional *xyz* e a intensidade de um determinado ponto;

- **PointXYZRGB** (Membros: *float x, y, z, rgb):* similar ao anterior, exceto o facto de conter a informação *rgb* de um determinado ponto;

- **Normal** (Membros: *float normal[3], curvature):* outro dos tipos de dados mais utilizados, representa a superfície normal e a medida de curvatura para um determinado ponto*;*

- **PointXYZRGBNormal** (Membros: *float x, y, z, rgb, normal[3], curvature):* estrutura que contém a informação tridimensional *xyz* e *rgb*, juntamente com a superfície normal e respetiva curvatura;

- **FPFHSignature33** (Membros: *float fpfh[33]):* tipo de ponto simples que armazena a informação *FPFH (Fast Point Feature Histogram)* de um determinado ponto [99].

3.4.2 Formato dos ficheiros PCD

O formato de ficheiros *PCD* (*Point Cloud Data*), nativo da *PCL*, não é o primeiro formato que suporta dados tridimensionais de nuvens de pontos. Este formato de ficheiro foi desenvolvido no sentido de complementar os formatos de ficheiros existentes que por alguma razão não suportam determinadas extensões que a *PCL* contém para o processamento de nuvens de pontos. As comunidades de computação gráfica e geométrica criaram inúmeros formatos que suportam a descrição de polígonos arbitrários e nuvens de pontos. Alguns dos formatos mais conhecidos incluem:

- ***OBJ*** – formato de ficheiro que suporta dados geométricos, desenvolvido pela *Wavefront Tecnologies*
- ***PLY*** (*PoLYgon file format*) – formato de ficheiro que contem dados poligonais, desenvolvido na *Universidade de Stanford*
- ***STL (STereoLithography)*** – formato de ficheiro nativo de software estereolitográfico *CAD*, criado pela *3D Systems*

Todos os formatos de ficheiros acima referidos possuem várias deficiências, isto porque foram desenvolvidos antes de a maioria das tecnologias e dos algoritmos de deteção terem sido inventados. Nenhum destes formatos anteriormente mencionados oferece a flexibilidade e rapidez dos ficheiros *PCD*. A capacidade de armazenar e processar conjuntos de nuvens de pontos organizados, diferentes tipos de dados, dados de histogramas *n*-dimensionais provenientes de descritores de características *3D* (*features*) e a utilização de métodos *mmap/munmap* para leitura e escrita de dados no disco são algumas das vantagens deste tipo de ficheiro relativamente os seus antecessores. Cada ficheiro *PCD* contém um cabeçalho que identifica e declara certas propriedades dos dados da nuvem de pontos armazenada no ficheiro (*Figura 36*) [100]. Estas propriedades podem ser descritas por:

- **VERSION:** especifica a versão do ficheiro *PCD*;
- **FIELDS:** especifica o nome de cada dimensão/campo que a *point cloud* pode ter;
- **SIZE:** especifica o tamanho de cada dimensão em bytes;
- **TYPE:** especifica o tipo de cada dimensão;
- **COUNT:** especifica o número de elementos em cada dimensão;
- **WIDTH/HEIGHT:** especifica a largura e a altura de uma *point cloud*;
- **VIEWPOINT:** especifica o ponto de aquisição do dispositivo que adquiriu a *point cloud;*
- **POINTS:** especifica o número total de pontos de uma nuvem;
- **DATA:** especifica o tipo de armazenamento que foi usado para guardar a point cloud;

```
# .PCD v.7 - Point Cloud Data file format
VERSION .7
FIELDS x y z rgb
SIZE 4 4 4 4
TYPE F F F F
COUNT 1 1 1 1
WIDTH 213
HEIGHT 1
VIEWPOINT 0 0 0 1 0 0 0
POINTS 213
DATA ascii
0.93773 0.33763 0 4.2108e+06
0.90805 0.35641 0 4.2108e+06
0.81915 0.32 0 4.2108e+06
0.97192 0.278 0 4.2108e+06
0.944 0.29474 0 4.2108e+06
0.98111 0.24247 0 4.2108e+06
```

Figura 36: Exemplo de um ficheiro *PCD* [100]

3.5 Filtragem e Segmentação

A nuvem de pontos inicial, retornada pelo sensor, geralmente necessita de algum tipo de filtragem e segmentação antes que possa ser utilizada pelos algoritmos de processamento. Normalmente, isto é devido à baixa precisão do sensor, o que leva a nuvens de pontos ruidosas, medições incorretas, buracos nas superfícies, entre outros. Por outro lado, a dimensão das nuvens de pontos pode ser tão elevada que o processamento levaria imenso tempo. Para solucionar estes problemas existem vários métodos de filtragem e segmentação capazes de evitar alguns destes erros recorrentes [101]. Alguns desses métodos são descritos nesta secção.

3.5.1 Filtro PassThrough

O filtro *PassThrough* permite a remoção de pontos de uma dimensão consoante um intervalo delimitador [k_{min}, k_{max}], definido pelo utilizador . Este filtro pode ser útil para descartar objetos desnecessários presentes no conjunto de dados ou restringir uma nuvem de pontos a um campo particular [101].

3.5.2 Remoção de Outliers

Os dispositivos de digitalização *3D* normalmente geram nuvens de pontos com diferentes densidades. Para além disso, os erros de medição podem levar a pontos dispersos (*outliers*) que corrompem os resultados ainda mais. A existência de *outliers* complica a estimativa das características locais de nuvens de pontos, tais como mudanças de curvaturas e normais da superfície, levando a valores errados, que por sua vez podem causar falhas de registo de nuvens de pontos [102]. Para evitar irregularidades nos conjuntos de dados podem ser implementados métodos de remoção de outliers, como os seguintes:

- ***Statistical Outlier Removal:*** corrige essas irregularidades calculando a média μ e desvio padrão σ das distâncias dos vizinhos mais próximos, e removendo os pontos que não satisfazem a condição $\mu \pm \alpha * \sigma$. O valor α depende do tamanho da vizinhança analisada [102], [103];
- ***Radius Outlier Removal:*** método de remoção de *outliers* mais simples. Elimina as irregularidades através do cálculo de pontos vizinhos que se encontram dentro de um raio *r*. A partir de um valor mínimo de vizinhos pré- definido, um ponto é considerado pertencente à nuvem de pontos se essa condição for cumprida. Se a condição não se verificar é considerado *outlier* [104].

3.5.3 Redução da resolução

Um sensor *Kinect* produz uma nuvem, com 307.200 (*640x480*) pontos, isto representa um elevado número de informações para serem processadas. A redução da resolução da nuvem de pontos *3D* (também denominada *downsampling)* é feita através de voxelização. Para tal, a nuvem de pontos é dividida em várias regiões em forma de cubo com a resolução desejada (*voxel grid*). Um *voxel* é um termo que está associado a um píxel *3D*, ou seja um cubo. Se o número de pontos dentro do cubo for superior a um determinado valor especificado, então todos os pontos são processados para que apenas um permaneça. A abordagem mais eficiente consiste em calcular o centroide, que é o ponto no qual as suas coordenadas são os valores médios de todos os pontos que pertencem ao *voxel*. Se este processo for feito com parâmetros sensíveis, a redução da resolução irá produzir resultados que são precisos o suficiente para trabalhar, o que traduz num menor custo computacional no processamento da nuvem de pontos resultante [101], [105].

3.5.4 Segmentação de planos

A segmentação de uma nuvem de pontos consiste na sua divisão em diferentes partes ou secções, sendo estes grupos de pontos denominados *clusters.* A ideia é dividir a nuvem de pontos para que as diferentes partes possam ser processadas independentemente. Idealmente, cada *cluster* deve representar um objeto diferente de uma cena. Para além disso, é possível segmentar certas formas específicas, tais como planos e cilindros [106].

No âmbito desta dissertação, é necessário a segmentação de diversas superfícies, com grande ênfase em estruturas planares, visto representarem grande parte da informação contida na nuvem de pontos. O primeiro grande problema consiste em encontrar o plano no conjunto de dados que suporta os objetos. Para acelerar o processo de pesquisa, o algoritmo irá fazer uso do método *RANdom SAmple Consensus* (*RANSAC*) para estimar os parâmetros do plano. O algoritmo pode ser descrito resumidamente pelas seguintes etapas:

1. Selecionar aleatoriamente três pontos únicos não-colineares $\{p_i, p_j, p_k\}$ de *P;*
2. Calcular os coeficientes do modelo dos três pontos ($ax + by + cz + d = 0$);
3. Calcular as distâncias de todos os pontos $p \in P$ ao plano $ax + by + cz + d = 0$;
4. Contar o número de pontos $p^* \in P$ cuja distância *d* ao plano se situa entre $0 \leq |d| \leq |d_t|$, onde d_t representa um valor especificado pelo utilizador.

Cada conjunto de pontos p^* é armazenado, e os passos mencionados anteriormente são repetidos para *k* iterações, onde k pode ser estimado utilizando:

$$k = \frac{\log(1-\alpha)}{\log(1-(1-\epsilon)^S)} \tag{3.1}$$

em que ϵ é a probabilidade de selecionar uma amostra que produz um má estimativa (ou seja, outlier), *S* é o número de pontos escolhidos para estimar os parâmetros do modelo [105], [107].

A vantagem da utilização do método *RANSAC* reside na sua capacidade de fazer uma estimativa robusta dos parâmetros do modelo mesmo com a presença de um número significativo *de outliers* na nuvem de pontos. Contudo, o tempo necessário para calcular estes parâmetros não pode ser quantificável, ou seja, quando o número de iterações é limitada a solução obtida pode não ser a ideal. Por outro lado, este método só pode estimar um modelo para uma determinada nuvem de pontos [108].

3.6 Deteção de pontos de interesse

Os *keypoints* (também referidos como pontos de interesse) são pontos numa imagem ou nuvem de pontos que são estáveis, distintos, e podem ser identificados através de um critério de deteção bem definido [83]. Não há uma definição rigorosa para o que constitui um bom método de deteção de pontos de interesse, no entanto este necessita de possuir as seguintes características:

1. **Dispersão entre pontos:** apenas um pequeno subconjunto de pontos na cena são pontos de interesse;
2. **Distinção entre pontos:** a área em redor de cada ponto de interesse deve ter uma forma ou aparência única;
3. **Repetibilidade:** se um ponto foi determinado como sendo um ponto de interesse de uma nuvem de pontos, esse ponto de interesse também deve ser encontrado no local correspondente numa nuvem de pontos semelhante.

Os métodos de deteção de pontos de interesse são vantajosos porque permitem reduzir o custo computacional associados à procura dos descritores de características (apenas operam sobre os pontos de interesse, logo tornam o processo mais eficaz) e ainda possibilitam a redução

do erro de correspondência entre pontos (evita a ambiguidade de correspondências em pontos que não sejam de interesse) [109].

Nesta secção dois algoritmos de deteção de pontos de interesse são descritos: *SIFT* (*Scale Invariant Feature Transform*) *Keypoints* e *Harris Keypoints*. Na documentação da biblioteca *PCL* é referido que ambos os métodos são adaptados de algoritmos de visão por computador que operam sobre imagens *2D*. No entanto, o funcionamento destes métodos em nuvens de pontos *3D* não se encontra devidamente documentada na biblioteca *PCL*.

3.6.1 SIFT Keypoints

O método *SIFT* (*Scale Invariant Feature Transform*) proposto por *David G. Lowe* [110] é um algoritmo desenvolvido para sistemas *2D* de visão por computador. Os descritores de características (*features*) detetados são altamente distintos e estáveis, com a vantagem de serem invariantes à escala, à rotação da imagem e à iluminação global. Existem muitos projetos de investigação que pretendem transferir o conceito para o espaço tridimensional, tais como os trabalhos [111], [112]. Na biblioteca *PCL*, este algoritmo é utilizado como módulo de deteção de pontos de interesse (*keypoints*) [83]. Esta implementação adapta o algoritmo original a partir de imagens para nuvem de pontos. Como mencionado anteriormente, por falta de documentação referente a este algoritmo no sistema *3D* por parte da biblioteca *PCL*, o método *SITF* será descrito resumidamente sob o ponto de vista de uma imagem *2D*.

Segundo *David G. Lowe* [110], o algoritmo *SIFT* pode ser dividido em 4 etapas principais:

- Deteção de extremos no espaço de escala;
- Localização dos pontos de interesse;
- Definição da orientação dos pontos de interesse;
- Descrição dos pontos de interesse.

As duas primeiras etapas descrevem a parte do detetor e as duas seguintes correspondem a formação do descritor. No âmbito da deteção dos pontos de interesse, as duas primeiras etapas são fundamentais para compreender a implementação deste algoritmo na *PCL*.

A primeira etapa da técnica *SIFT* consiste em identificar os pontos que são invariantes a mudanças de escala e a diferentes posições de visualização do mesmo objeto. Tal objetivo é alcançado através da procura de características estáveis em diferentes escalas, recorrendo a uma

função de espaço de escala (função *Gaussiana*). Uma imagem $I\ (x, y)$ passa a ser definida por $L\ (x, y, \sigma)$ no espaço de escala. Esta função é obtida pela convolução de uma função Gaussiana, $G(x, y, \sigma)$ com a imagem $I\ (x, y)$ (*equação 3.2*) [110], [113].

$$L\ (x, y, \sigma) = G\ (x, y, \sigma) * I\ (x, y) \tag{3.2}$$

em que,

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma} e^{-(x^2+y^2/2\sigma^2)} \tag{3.3}$$

A eficiência da deteção dos pontos de interesse é aumentada com a utilização de uma função *DoG* (*Difference of Gaussian*) formada pela diferença de imagens filtradas em escalas próximas, separadas por uma constante de escala *k*. A função *DoG* é definida por:

$$\begin{aligned} DoG\ (x, y, \sigma) &= \big(G(x, y, k\sigma) - G(x, y, \sigma)\big) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned} \tag{3.4}$$

Esta função *DoG* tem como objetivo suavizar as imagens, por forma a obter imagens onde os detalhes indesejados e ruído são eliminados, reforçando as características desejáveis. Com a variação do valor σ é possível encontrar tais características em diferentes escalas. O próximo passo envolve a deteção de extremos em cada intervalo de cada oitava. Um extremo é definido como qualquer valor de *DoG* maior do que todos os seus vizinhos no espaço de escala.

Para detetar os valores de máximo ou mínimo locais da função *DoG*, a intensidade de cada ponto é comparada com as intensidades dos seus 8 vizinhos na sua escala e com 9 pontos vizinhos da escala superior e inferior (*Figura 37*) [110], [113].

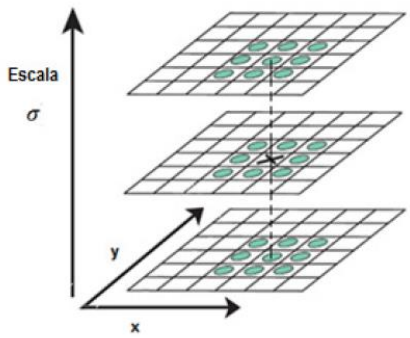


Figura 37: Deteção de extremos no espaço de escala [110]

Se este valor é mínimo ou máximo de todos os pontos testados, então este ponto pode ser considerado um extremo, ou seja um candidato a ponto de interesse. A próxima etapa consiste na identificação da localização exata destes pontos detetados como extremos. A partir de uma expansão de *Taylor* da função *DoG* aplicada à imagem, $DoG\ (x, y, \sigma)$, deslocada de modo a que a origem desta expansão esteja localizada no ponto de amostragem, é possível determinar uma localização interpolada desse extremo (*equação 3.6*) [110], [113].

$$D(\bar{x}) = D + \frac{\partial D^T}{\partial \bar{x}}\bar{x} + \frac{1}{2}\bar{x}^T\frac{\partial^2 D}{\partial x^2}\bar{x} + \cdots \qquad (3.5)$$

em que $\bar{x} = (\mathrm{x}, \mathrm{y}, \sigma)^T$

A localização em sub-píxeis do ponto de interesse é dada pelo extremo da função apresentada na *equação 3.5*. Esta localização, $\hat{x}$, é determinada ao se calcular a derivada de $D(\bar{x})$, em relação a x , e igualando o resultado a zero. Portanto, a posição do extremo é dada por:

$$\hat{x} = -\frac{\partial^2 D^{T-1}}{\partial \bar{x}^2}\frac{\partial^2 D}{\partial \bar{x}} \qquad (3.6)$$

O valor da função no extremo, $D(\bar{x})$, é útil para a rejeição de pontos com baixo contraste (instáveis), que seriam sensíveis ao ruído. Segundo *David G. Lowe* é aconselhável rejeitar os valores de $|D(\bar{x})|$ inferiores a um determinado limiar. Além deste procedimento para eliminar os pontos instáveis, a função *DoG* possui uma forte resposta ao longo das arestas, isto é, os pontos em arestas poderiam ser escolhidos como pontos de interesse, o que é indesejável. Para solucionar este problema, a deteção e eliminação destes pontos pode ser feita recorrendo a uma matriz *Hessiana* 2x2 [110], [113].

3.6.2 Harris Keypoints

O método proposto por *Chris Harris e Mike Stephens* [114], o *Harris Keypoints*, é um algoritmo desenvolvido para detetar cantos e extremidades em sistemas *2D* de visão por computador. Este método é caracterizado pela elevada mudança de intensidade nas direções horizontal e vertical. Esta característica pode ser utilizada na análise da forma e de movimento, diretamente detetadas a partir de imagens em tons de cinza (*grayscale*) [115]. Para o caso em

3D, a implementação na biblioteca *PCL* substitui os gradientes da matriz de covariância pelas normais da superfície. Assim como no método anterior, este algoritmo é utilizado na biblioteca *PCL* como módulo de deteção de pontos de interesse (*keypoints*) [83]. Como a documentação na biblioteca *PCL* acerca deste algoritmo de deteção é reduzida, o método *Harris* será descrito resumidamente sob o ponto de vista de uma imagem *2D*.

Harris e Stephens, propuseram um método de deteção de pontos de interesse para imagens que se tornou muito conhecido devido à sua forte invariância à rotação, escala, variação de iluminação e ruído presente na imagem. Este método é baseado numa função local de auto-correlação de um sinal, que mede as alterações locais do sinal devido a pequenos deslocamentos de uma janela em diferentes direções [116].

A auto-correlação local é definida como:

$$e\ (x,y) = \sum_{x_i,\, y_i} W(x_i,\ y_i)[I(x_i + \Delta x, y_i + \Delta y) - I(x_i, y_i)]^{\,2} \qquad (3.7)$$

em que *I* representa a função da imagem e $(x_i,\ y_i)$ são os pontos da função Gaussiana W, centrada no ponto (x, y), que define a área da vizinhança em análise.

Através da utilização de uma expansão de *Taylor* truncada para os primeiros termos da ordem para aproximar a imagem deslocada, é obtido o seguinte:

$$\begin{aligned} e\ (x,y) &= S \begin{bmatrix} \sum_{x_i, y_i} W.I_x^2 & \sum_{x_i, y_i} W.I_x.I_y \\ \sum_{x_i, y_i} W.I_x.I_y & \sum_{x_i, y_i} W.I_y^2 \end{bmatrix} S^T \\ &= S.E(x,y).S^T \end{aligned} \qquad (3.8)$$

onde $S = [\Delta x, \Delta y]$ é o vetor de deslocamento, I_x e I_y denotam as derivadas parciais de *x* e *y*, respetivamente, e juntamente com W são avaliadas em $(x_i,\ y_i)$ pontos.

Harris e Stephens propuseram analisar os valores próprios da matriz *E*, que contem informação local suficiente relacionado com a estrutura da vizinhança (pontos de interesse).

Além disso, para evitar um cálculo dispendioso de valores próprios, atribuíram a cada píxel da imagem o seguinte valor:

$$h(x, y) = \det(E) - k(tr(E))^{2} \qquad (3.9)$$

em que *k* é uma constante, *det* e *tr* são o determinante e o *trace (*função que soma os elementos diagonais de uma matriz n *x* n) da matriz E, respetivamente.

Este algoritmo tem sido utilizado em muitas aplicações no processamento de imagens e em sistemas de visão por computador devido a sua simplicidade e eficácia [116].

3.7 Deteção das características

Os descritores de características (*features)* são representações de um determinado ponto ou posição tridimensional no espaço, eu descrevem padrões geométricos com base nas informações disponíveis em torno do ponto [79]. Uma boa representação de uma característica de um determinado ponto difere de uma má representação, por ser capaz de capturar as mesmas características locais da superfície na presença de [107]:

1. **Transformações rígidas:** rotações e translações *3D* no conjunto de dados não devem influenciar a estimativa do vetor de características *F*;
2. **Densidade de amostras variável:** uma amostra da superfície mais ou menos densa deve ter a mesma assinatura no vetor de características;
3. **Ruído:** a representação da característica de um ponto deve manter os mesmos ou valores muito semelhantes no vetor de características na presença de ruído leve no conjunto de dados.

O *PFH* (*Point Feature Histograms*) e *FPFH* (*Fast Point Feature Histograms*) são os dois métodos mais amplamente utilizados para deteção de características geométricas de um ponto, sendo estimados a partir da curvatura e da nomal da superfície adjacente a um certo ponto. Ambos são considerados detetores de características locais, pois caracterizam um ponto utilizando a informação fornecida pelos seus vizinhos mais próximos. Para determinar a sua vizinhança de forma eficiente, o conjunto de dados é normalmente dividido em conjuntos menores usando técnicas de decomposição espaciais como *octrees* e *kdtrees* [79].

Nesta secção são descritos dois algoritmos de deteção de características geométricas locais, o *PFH* e o *FPFH*, e para complementar estes dois algoritmos é descrito inicialmente o método de estimação das normais da superfície (*Normal Estimation*).

3.7.1 Estimativa das normais da superfície

A estimação das normais da superfície é uma propriedade importante de uma superfície geométrica, e são vastamente utilizadas em aplicações de computação gráfica, por exemplo para determinar a orientação de um sistema de coordenadas ou para aplicar as fontes de luz corretas que originam sombras e outros efeitos visuais. O problema de determinar a normal a um certo ponto da superfície é aproximado pelo problema de estimação da normal de um plano tangente à superfície, que por sua vez representa um problema de mínimos quadrados da estimação do plano em P^k. O plano é representado por um ponto *x* e por um vetor normal $\vec{n}$, sendo a distância a partir do ponto $p_i \in P^k$ para o plano definida como $d_i = (p_i - x) * \vec{n}$. Os valores de *x* e $\vec{n}$ são calculados no sentido de mínimos quadrados, de modo a que $d_i = 0$, assumindo que:

$$x = \bar{p} = \frac{1}{k} * \sum_{i=1}^{k} p_i \qquad (3.10)$$

é o centroide de P^k , a solução para $\vec{n}$ é conhecida através da análise dos valores e vetores próprios da matriz de covariância $C \in \mathbb{R}^{3*3}$ de P^k, expressa da seguinte forma:

$$C = \frac{1}{k}\sum_{i=1}^{k} \xi_i * (p_i - \bar{p}) * (p_i - \bar{p})^T, C * \overrightarrow{v_j} = \lambda_j * \overrightarrow{v_j}, j \in \{0, 1, 2\} \qquad (3.11)$$

onde *k* é o número de pontos vizinhos considerados na vizinhança de p_i e $\vec{p}$ representa o centroide *3D* dos vizinhos mais próximos. λ_j e $\overrightarrow{v_j}$ é o valor próprio e vetor próprio do elemento j da matriz de covariância, respetivamente. O termo ξ representa um possível peso de p_i e, normalmente é igual a 1. Se $0 \leq \lambda_0 \leq \lambda_1 \leq \lambda_2$, o vetor próprio $\overrightarrow{v_0}$ corresponde ao valor próprio mais pequeno λ_0 , portanto é aproximadamente $+\vec{n} = \{n_x, n_y, n_z\}$ ou $-\vec{n}$ [107], [117].

3.7.2 PFH (Point Feature Histograms)

O objetivo do descritor *PFH* consiste em codificar as propriedades geométricas da vizinhança de um determinado ponto generalizando a curvatura em redor do ponto utilizando histogramas de valores multi-dimensionais [118]. A representação de um descritor *PFH* é baseada na relação entre os pontos da vizinhança e nas normais da superfície estimadas. A computação deste descritor é feita da seguinte forma [119]:

1. Para cada ponto p_q, todos os pontos vizinhos que se encontrem dentro de uma esfera de raio *r* são selecionados (vizinhança) (*Figura 38*);
2. Para cada par de pontos p_i e p_j $(i \neq j)$ dentro da vizinhança do ponto p_q, e as suas normais associadas n_i *e* n_j, é definido um sistema de coordenadas fixo para um dos pontos, assumindo que:
    a. $u = n_i$
    b. $v = (p_j - p_i) * u$
    c. $w = u * v$
3. Por fim, para cada par de pontos dentro da vizinhança é calculado as variações angulares de n_i *e* n_j da seguinte forma:
    a. $\alpha = v * n_j$
    b. $\emptyset = (u * (p_j - p_i))/\|p_j - p_i\|$
    c. $\theta = \arctan(w * n_j, u * n_j)$

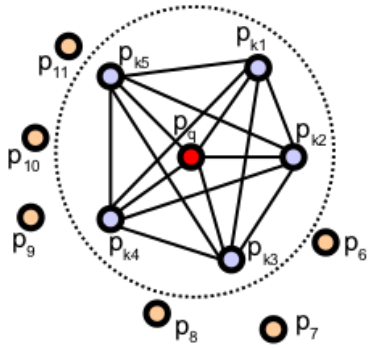


Figura 38: Diagrama da região de influência para o *PFH* [119]

3.7.3 FPFH (Fast Point Feature Histograms)

A complexidade computacional do método de deteção de características *PFH* para uma determinada nuvem de pontos P composta por n pontos é definida por $O(n * k^2)$, onde k representa o número de vizinhos para cada ponto p da nuvem de pontos P. Uma otimização simples deste processo consiste em armazenar os valores dos descritores de características (*features*) e reordenar o conjunto de dados da nuvem de pontos de forma que a realização de pesquisas num grupo de dados se torne mais rápida do que o processo de recalcular esses valores. Para aplicações em tempo real ou perto disso, o cálculo dos descritores das características recorrendo ao método *PFH* em vizinhanças densas pode representar um grande problema [119].

Este método, *FPFH* (*Fast Point Feature Histograms*), representa um modelo simplificado que reduz o custo computacional do algoritmo para $O(n * k)$, enquanto mantem a maior parte do poder discriminativo do método *PFH*. Para simplificar o processo de deteção de características, procede-se do seguinte modo:

1. Para cada ponto p_q, calcula-se as variações angulares $(\alpha, \emptyset, \theta)$ entre si e os seus vizinhos, tal como foi mencionado no método *PFH*. Este passo é denominado *SPFH* (*Simplified Point Feature Histogram*);
2. Para cada ponto, os seus vizinhos são recalculados, e os valores do *SPFH* são utilizados para pesar o histograma final de p_q (denominado *FPFH*), da seguinte forma:

$$FPFH\ (p_q) = SPFH\ (p_q) + \sum_{i-1}^{k} \frac{1}{\omega_k} * SPFH\ (p_k) \qquad (3.12)$$

onde o peso ω_k representa a distância entre o ponto p_q e um ponto vizinho p_k num dado espaço métrico. Para melhor compreender a importância deste sistema, a *Figura 39* representa o diagrama da região de influência de uma vizinhança a partir do ponto p_q[107], [119], [120].

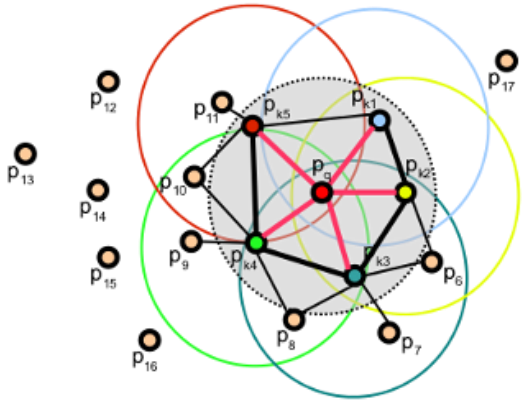


Figura 39: Diagrama da região de influência para o *FPFH* [107]

3.7.4 Diferenças entre PFH e FPFH

As principais diferenças entre o método *PFH* e *FPFH* são descritos de seguida:

- O método *FPFH* não consegue interligar todos os vizinhos do ponto p_q. A ausência de alguns destes pares pode influenciar a captura geométrica em torno do ponto p_q;
- Os modelos *PFH* são precisamente determinados em torno do ponto p_q, enquanto o *FPFH* inclui pares de pontos adicionais fora da esfera de raio *r* (no máximo *2r*);
- Devido ao esquema de reponderação, o *FPFH* combina os valores *SPFH* e volta a determinar os valores de alguns dos pares de pontos vizinhos [107].

3.8 Registo de nuvens de pontos

O problema de alinhamento de várias nuvens de pontos *3D* num modelo completo consistente é conhecido como registo (*registration*). O objetivo do registo de nuvens de pontos consiste na determinação das posições relativas e das orientações de um conjunto de nuvens *3D*, e na obtenção de um modelo global consistente, através da correta sobreposição das diversas áreas que se intersectam. Para um determinado conjunto de nuvens de pontos, adquiridas separadamente a partir de diferentes pontos de vista, é necessário encontrar um sistema que seja capaz de alinhar as diversas nuvens num único modelo. A ideia-chave passa por identificar os pontos correspondentes entre o conjunto de nuvens de pontos e encontrar uma transformação que minimiza a distância (erro de alinhamento) entre os pontos correspondentes. Este processo é repetido, uma vez que a pesquisa das correspondências é influenciada pela posição e orientação relativa dos conjuntos de dados. A partir do momento que o erro de

alinhamento é inferior a um determinado valor (*threshold*), o registo de nuvens de pontos está completo [84], [107], [121].

Um dos métodos mais populares para registo de nuvens de pontos é o algoritmo *Iterative Closest Point* (*ICP*). Infelizmente, este algoritmo tem algumas desvantagens, sendo a mais evidente a necessidade de um elevado número de etapas de iteração até que a convergência possa ser alcançada. Embora tenham sido apresentadas várias soluções, não existe uma única abordagem que possa resolver o problema na sua totalidade. Uma das possíveis soluções passa por determinar uma boa transformação inicial e retornar um conjunto de correspondências que podem ser utilizadas para transformar diretamente a nuvem de pontos de origem na área de convergência do alvo. No sentido mais geral, o objetivo pode ser formulado por:

1. **Alinhamento Inicial:** efetuar uma transformação à nuvem de ponto de origem, a partir de uma posição inicial arbitrária em relação ao modelo, de modo a alcançarmos um modelo suficientemente próximo do ideal;
2. **Alinhamento Refinado:** efetuar o refinamento da solução encontrada pelo algoritmo de alinhamento inicial, de modo a obtermos um resultado globalmente ideal.

Com base na qualidade das correspondências encontradas, as áreas de sobreposição das duas nuvens de pontos podem ser quase coincidentes após o alinhamento inicial. Esta solução representa um bom ponto de partida para algoritmos utilizados no refinamento, tais como o *ICP*, que outrora teriam sido questionados devido ao seu elevado custo computacional associado ao processo de registo [107].

3.8.1 Alinhamento Inicial

Segundo *David G. Lowe* [119] é descrito um método de alinhamento inicial baseado na formulação *SAmple Consensus*, denominado *SAC-IA* (*Sample Consensus Initial Alignment*). Este método tem como objetivo manter as mesmas relações geométricas das correspondências sem ter que tentar todas as combinações possíveis de um conjunto de correspondências. Para isso, é analisado um grande número de candidatos a correspondências, sendo cada uma delas rapidamente classificada utilizando a seguinte formulação:

1. Selecionar *s* amostras de pontos de *P*, enquanto a distância entre pares for maior do que a distância mínima definida pelo utilizador;

2. Para cada uma das amostras de pontos, é necessário encontrar uma lista de pontos em *Q* cujos histogramas são semelhantes aos histogramas dos pontos amostrados. A partir dessa lista, um ponto é selecionado ao acaso e considerado como a correspondência do ponto amostrado;
3. Calcular a transformação rígida definida pelas amostras de pontos e pelas suas correspondências. Para além disso, é necessário encontrar o erro métrico para a nuvem de pontos que calcula a qualidade da transformação.

Durante este processo, estas três etapas são repetidas várias vezes. A transformação que obteve melhor erro métrico é armazenada e utilizada para alinhar aproximadamente os pontos de vista parciais. Por último, uma otimização não linear local é aplicada recorrendo ao algoritmo de *Levenberg-Marquardt* [119].

Na biblioteca *PCL*, este algoritmo de alinhamento inicial não é o único implementado [84]. No módulo de registo é possível ainda encontrar outros métodos capazes de realizar a estimação da transformação *3D* de nuvens de pontos, tais como o método *TransformationEstimationSVD* [122] e *TransformationEstimationPointToPlaneLLS* [123]. No primeiro caso, a estimativa da transformação é baseada no algoritmo *SVD* (*Singular Value Decomposition*), sendo o alinhamento realizado através dos valores das correspondências obtidas por outros processos. No segundo caso, o algoritmo é desenvolvido com recurso a uma aproximação linear, denominada *LLS* (*Linear Least Squares*), para minimizar a distância entre um ponto e um plano dos pontos correspondentes de duas nuvens de pontos recorrendo à utilização das normais da superfície.

3.8.2 Alinhamento Refinado

Quando o alinhamento inicial é concluído, o próximo processo envolve o refinamento do resultado obtido, que é realizado pelo algoritmo *ICP* (*Iterative Closest Point*). O *ICP* é um algoritmo iterativo utilizado para encontrar a transformação ideal de duas nuvens de pontos através minimização da diferença da distância entre as áreas que se sobrepõem [107]. Este algoritmo pode ser generalizado através das seguintes etapas:

1. Associar os diversos pares de pontos (correspondências) entre as duas nuvens de pontos;
2. Estimar a transformação (rotação *R* e translação *t*) utilizando uma função de mínimos quadrados das distâncias entre os pontos correspondentes p_i e q_i:

$$\min \sum_{i=1}^{n} dist(R * q_i + t, p_i) \qquad (3.13)$$

3. Aplicar a transformação estimada à nuvem de pontos que está a ser registada;
4. Repetir o processo até que um dos critérios de terminação seja atingido [107], [124].

Contudo, a implementação do método linear *ICP* na biblioteca *PCL* utiliza o algoritmo *SVD* (*Singular Value Decomposition*) para encontrar a solução de mínimos quadrados para a matriz de rotação e para o vetor de translação entre o conjunto de pontos correspondentes [125]. Para além desta formulação, existem ainda outras duas que derivam do *ICP* linear: o *ICP* não linear [126], baseado no método de otimização *Levenberg-Marquardt* e uma variação do *ICP* que se baseia na informação das normais da superfície de um plano para estimar a transformação [127]. Uma característica comum a estas três implementações é o facto de todas possuírem vários critérios de terminação. Estes critérios são descritos de seguida:

- O número de iterações atingiu o máximo pré-estabelecido (critério 1);
- A diferença entre a transformação anterior e a atual estimada é menor que um valor pré-determinado (critério 2);
- A soma do erro *Euclideano* dos quadrados é menor que um valor pré-determinado (critério 3) [125].

3.9 Reconstrução da superfície

Os dados obtidos a partir do registo de múltiplas nuvens de pontos não são perfeitos, podendo conter algumas irregularidades causadas por pequenos erros de medida de distância, sendo dificilmente removidos por meio de análise estatística. Por outro lado, devido ao facto de uma área sobreposta de duas nuvens de pontos ser quase coincidente, a densidade de pontos nessa área é aproximadamente duas vezes maior que o resto do conjunto de dados. Esta densidade de pontos variável terá um efeito negativo na estimação das curvaturas ou das normais superfície. Para além disso, para a criação de modelos completos devem ser contabilizados os erros de captura, provenientes de superfícies brilhantes, objetos metálicos, bem como erros de oclusão. A solução passa pela utilização de algoritmos para o processamento de superfícies, que sejam capazes de recriar as partes da superfície em falta e suavizar as regiões

com excesso de informação, de forma a corrigir estes pequenos erros e a obter um modelo globalmente consistente [107].

Na reconstrução de modelos tridimensionais, o modelo global obtido a partir de múltiplas digitalizações adquiridas por um equipamento de deteção *3D* encontra-se no formato de nuvem de ponto. Este modelo apenas revela uma aproximação da superfície, sendo que as informações mais detalhadas, tais como a área da superfície do objeto registado, não são fornecidas. Para uma melhor análise qualitativa do modelo *3D* é necessário um processo de reconstrução de malha tridimensional, denominada *mesh*. Normalmente, esta malha *3D* é constituída por diversos triângulos conectados entre si, sendo estes a unidade básica de construção de qualquer polígono [128].

O problema da reconstrução da superfície envolve vários algoritmos que podem ser divididos em 4 grupos consoante o objetivo para o qual foram desenvolvidos:

- **Processamento da superfície da nuvem de ponto:** obter uma melhor aproximação da superfície (por exemplo, algoritmo *Moving Least Squares*);
- **Construção de uma malha (*mesh*):** converter a nuvem de pontos numa malha sem modificar a posição dos vértices (tais como: *ConvexHull*, *Greddy Triangulation)*;
- **Reconstrução da superfície:** criação de uma malha a partir de uma nuvem de pontos com a possibilidade de modificação da posição dos vértices (por exemplo, *Poisson*, *Marching Cubes, GridProjection*);
- **Processamento da malha (*mesh*):** aperfeiçoamento das malhas criadas através da modificação das ligações e/ou vértices (tais como: *MeshSmoothingLaplacianVTK*) [129].

Nesta secção são descritos resumidamente alguns dos algoritmos mencionados anteriormente, utilizados para a reconstrução e processamento de superfícies e de malhas *3D*. Estes algoritmos encontram-se implementados no módulo *surface* da biblioteca *PCL* [87].

3.9.1 Moving Least Squares (MLS)

O método *MLS* (*Moving Least Squares*), desenvolvido por *David Levin* [130], é uma das diversas técnicas de reconstrução da superfície. Este método proporciona uma interpolação da superfície para um determinado conjunto de pontos *P* por encaixe de polinómios bivariáveis de ordem superior para cada ponto da sua vizinhança. Em contraste com outras técnicas de

interpolação ou reamostragem, o algoritmo *MLS* tem a vantagem de que a superfície resultante é deduzida a partir da informação da nuvem de pontos original. Em aplicações *3D*, é utilizado para criar superfícies tridimensionais com base na redução (*downsampling*) e no aumento (*upsampling*) da resolução de uma nuvem de pontos (*Figura 40*) [107].

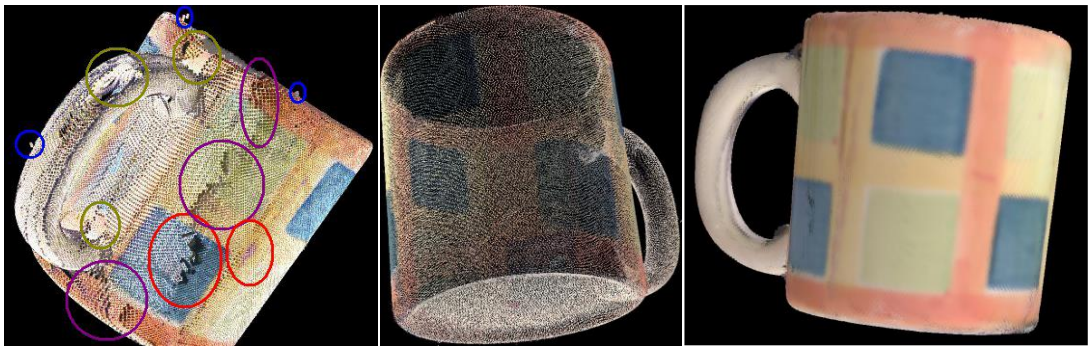


Figura 40: Exemplo de uma nuvem de pontos com diversos erros (à esquerda). Aplicação de uma redução da resolução, *downsampling* (ao centro) e um aumento da resolução, *upsampling* (à direita) à nuvem de pontos inicial [107]

3.9.2 Greedy Triangulation (GT)

O algoritmo *Greedy Triangulation* (*GT*) de um conjunto *S* de *n* pontos de um plano é representada pela triangulação obtida a partir de um conjunto vazio e, posteriormente em cada passo ir adicionando repetidamente o menor lado compatível entre dois pontos. Um lado compatível é definido como um lado que não atravessa nenhuma das arestas adicionadas anteriormente. Após terem sido processados todos os pontos, o resultado que se obtém é uma malha composta por um conjunto de triângulos que representa a nuvem de pontos inicial (*Figura 41-a*) [131].

3.9.3 Marching Cubes (MC)

O algoritmo *Marching Cubes* (*MC*), proposto por *William Lorensen* e *Harvey Cline* [132], é um método que se baseia na informação de um *voxel* para extrair a superfície isométrica. A cada *voxel* é atribuído um valor, e o algoritmo tenta criar uma malha para a superfície com um valor isométrico especificado. A partir da informação dos limites de cada *voxel*, é feita uma comparação com os 15 diferentes casos pré-definidos. A configuração da malha correspondente para o *voxel* pesquisado é adicionada no seu lugar. Após terem sido processados todos os *voxels*, o resultado é uma malha que se aproxima da forma da nuvem de pontos a partir do qual foi criada (*Figura 41-b*) [133].

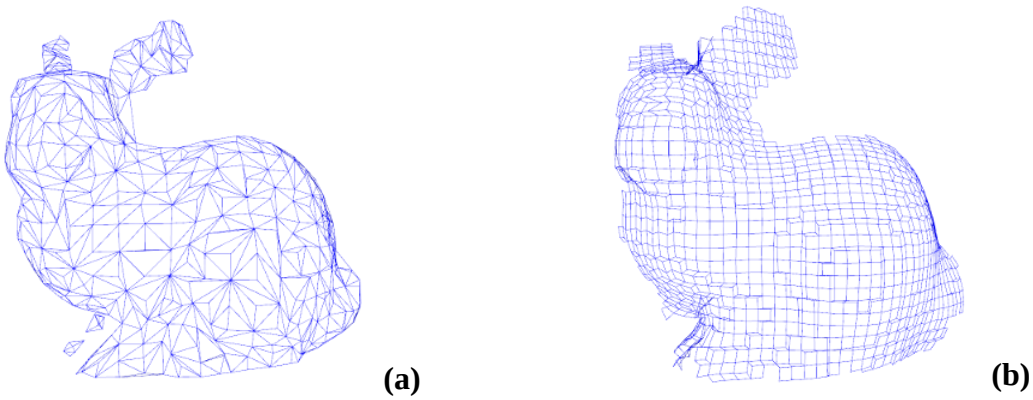


Figura 41: Resultado da reconstrução da superfície de um objeto por dois métodos diferentes: *Greedy Triangulation* (a) e *Marching Cubes* (b) [87]

4. CONSTRUÇÃO DO MODELO TRIDIMENSIONAL

Este capítulo apresenta a metodologia utilizada para a implementação do scanner 3D baseado no sensor Microsoft Kinect. A configuração das técnicas e dos métodos aplicados para a captura e processamento nuvens de pontos, bem como para a visualização do modelo tridimensional obtido são explicadas detalhadamente neste capítulo.

4.1 Arquitetura do software

O foco da implementação deste projeto é baseada na aquisição de dados a partir do sensor *Microsoft Kinect*. Para tal, é necessário um conjunto de bibliotecas *open-source* e *frameworks* capazes de obter, processar e reconstruir os dados obtidos por este sensor. Como referido no capítulo anterior, a utilização dos drivers da *PrimeSense* e do *framework OpenNI* permite a interação com este dispositivo. A biblioteca *PCL* foi utilizada para o processamento e reconstrução das nuvens de pontos *3D*. Nesta dissertação, a linguagem de programação utilizada foi o *C++* projetada na plataforma de desenvolvimento *Code::Blocks*.

4.2 Metodologia

Para alcançar o objetivo de construção de um modelo tridimensional através de múltiplas capturas de nuvens de pontos de um determinado objeto é necessário um conjunto de etapas complexas que respondem a uma diversidade de problemas. Para solucionar de forma prática estes problemas, as referidas etapas podem ser divididas em 5 grandes módulos: aquisição, pré-processamento, processamento, registo e reconstrução de nuvens de pontos. Estes módulos podem ser descritos pelas seguintes fases:

1. **Aquisição das nuvens de pontos:**
    - **a.** Obtenção da informação *3D* de um objeto;
    - **b.** Especificação da região de interesse;
    - **c.** Utilização de técnicas de segmentação de objetos;
2. **Pré-processamento das nuvens de pontos:**
    - **a.** Aplicação de métodos de transformação e filtragem;
    - **b.** Estimativa das normais da superfície;

3. **Processamento das nuvens de pontos:**
    a. Deteção dos pontos de interesse (*keypoints*);
    b. Calculo dos descritores de características (*features*);
    c. Procura de um conjunto de correspondências;
    d. Remoção das más correspondências desse conjunto**;**
4. **Registo das nuvens de pontos:**
    a. Estimação da transformação rígida (alinhamento inicial);
    b. Aperfeiçoamento da transformação estimada (alinhamento refinado);
5. **Processamento/Reconstrução da superfície:**
    a. Pré-processamento da superfície do modelo
    b. Reconstrução da superfície do modelo.

4.3 Aquisição da informação 3D

A maior parte da informação utilizada para a digitalização de um objeto é adquirida a partir da câmara de profundidade do sensor *Microsoft Kinect*. A representação de dados mais frequente é uma imagem *RGB-D*, isto é, uma combinação dos valores dos três canais de cor (*red – R, green – G, blue – B*) com a informação de profundidade (*depth - D*). Através do *framework OpenNI*, a imagem *RGB* e de profundidade podem ser combinadas por forma a produzir uma representação de pontos (nuvem de pontos), constituída por 307.200 pontos individuais cada um com as coordenadas *x, y, z* e os valores *RGB*. A *Figura 42* ilustra uma nuvem de pontos de um ambiente *indoor* adquirida a partir do sensor *Microsoft Kinect.*

Figura 42: Nuvem de pontos adquirida pelo sensor *Kinect*

4.3.1 Calibração do sensor *Kinect*

Como o sensor infravermelho está posicionado a alguns centímetros da câmara *RGB*, um píxel na imagem *RGB* refere-se a um ponto diferente do mesmo píxel da imagem de profundidade. Para corrigir esta diferença de posição entre as duas câmaras é necessário calibrar estas câmaras. O *OpenNI* vem com uma calibração pré-definida que alinha diretamente a imagem de profundidade e de cor com um comprimento focal virtual constante. Este *framework* permite a utilização de diversas aplicações sem necessidade de qualquer passo adicional de calibração, no entanto, em algumas aplicações robóticas pode ser necessário uma calibração mais precisa.

Embora o sensor *Kinect* seja capaz de proporcionar uma boa resolução para a reconstrução *3D* de objetos, o seu sensor de profundidade tem problemas com o ruído ótico, superfícies brilhantes e áreas oclusivas. A *Figura 43* mostra um erro na aquisição de cor nos limites do objeto e a visualização da área oclusiva por trás do objeto.

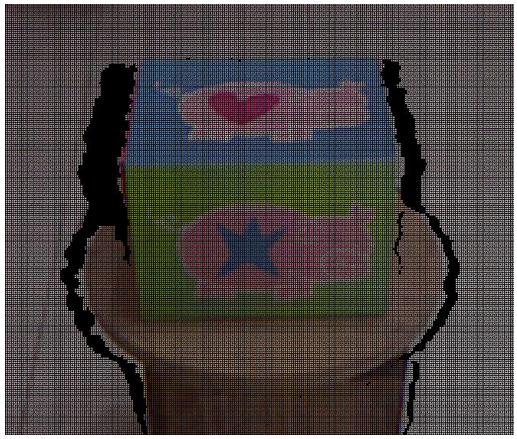


Figura 43: Erro na aquisição da cor de um objeto

O erro de cor nas extremidades do objeto é explicado pela pequena diferença entre a imagem de profundidade e *RGB* que o sensor providencia. Esta mudança é única para cada dispositivo *Microsoft Kinect*, e só poderia ser superada através da implementação de um método de calibração. Neste projeto de investigação, nenhuma ferramenta de calibração adicional foi utilizada com o objetivo de corrigir este problema.

4.3.2 Sistema de coordenadas do sensor

O principal objetivo de um sistema de coordenadas é especificar como o sensor interpreta o ambiente e produz os dados tridimensionais. O sistema de coordenadas que se deverá ter em conta é o sistema de coordenadas do sensor. Para expressar as coordenadas *3D* de um objeto é considerado um sistema de coordenadas de profundidade com a origem no centro da câmara infravermelha. Assim sendo, quando se segura no sensor horizontalmente e se aponta em direção a uma determinada cena ou objeto, o eixo *x* positivo estende-se para a direita, o eixo *y* positivo aponta para o chão e o eixo *z* positivo desloca-se para dentro da cena. Este sistema de coordenadas associado ao sensor infravermelho encontra-se ilustrado na *Figura 44*.

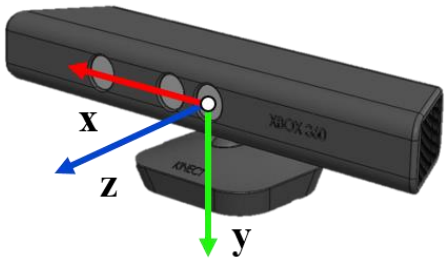


Figura 44: Sistema de coordenadas do sensor *Microsoft Kinect*

No entanto, o sistema de coordenadas utilizado por defeito no *framework OpenNI* é ilustrado na *Figura 45*. Para converter a nuvem de pontos, é necessário traduzir as coordenadas projetivas que correspondem a imagem de profundidade tridimensional, para o sistema de coordenadas definido pelo *framework OpenNI*. Para isso, na fase de pré-processamento é utilizada uma matriz *4x4* para manipular através de uma transformação *3D* a nuvem de pontos original. Contudo, para uma visualização direta da nuvem de pontos capturada pelo sensor *Kinect*, o sistema de coordenadas pode ser ajustado utilizando funções implementadas na classe *pcl::visualization::PCLVisualizer* presente na biblioteca *PCL*.

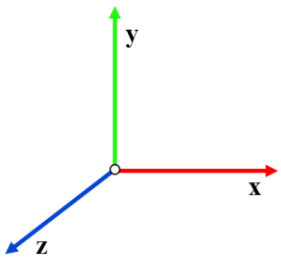


Figura 45: Sistema de coordenadas definido pelo *OpenNI*

4.3.3 Posicionamento do sensor

O sensor *Microsoft Kinect* tem de ser posicionado por forma a que se consiga capturar a face superior bem como a superfície frontal de um determinado objeto ao mesmo tempo. Para isso, o sensor tem de ser colocado num nível mais elevado relativamente ao objeto que se pretende digitalizar. Contudo, o ângulo deste sensor tem de ser ajustado para um valor inferior em relação ao ângulo normal deste sensor quando se encontra na horizontal. Estes ajustes são necessários para que o sistema possa alcançar melhores resultados e ao mesmo tempo seja possível adaptar-se as dimensões dos diferentes objetos que se pretende digitalizar. A *Figura 46* ilustra a configuração que garante o melhor funcionamento do sistema para um determinado conjunto de objetos.

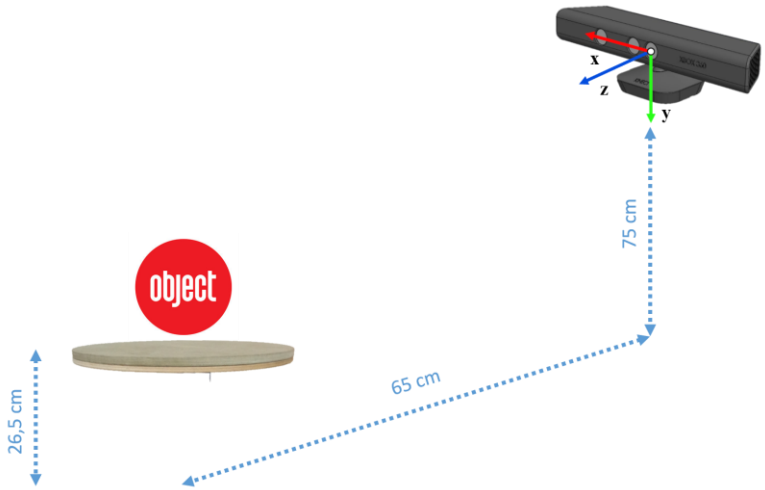


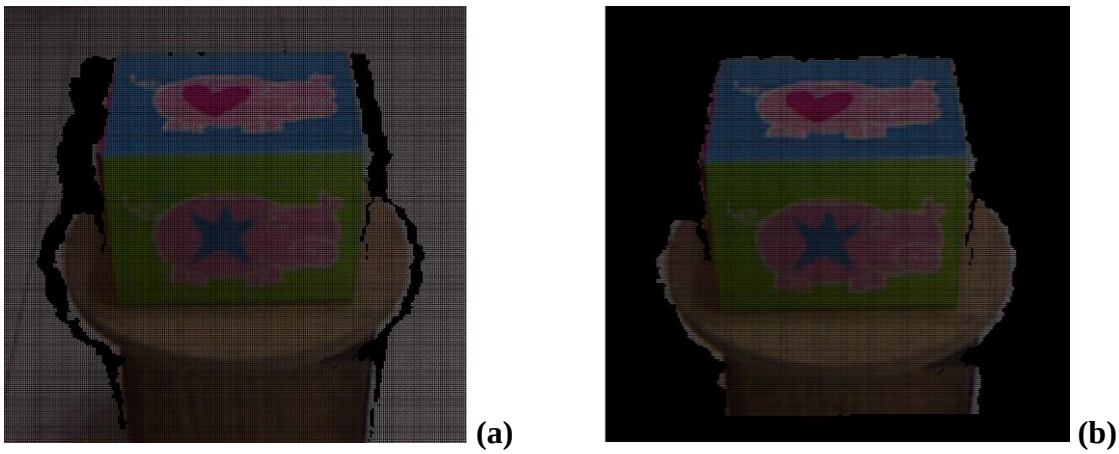
Figura 46: Posicionamento do sensor em relação à plataforma rotativa

O ângulo de inclinação do sensor pode ser ajustado entre -27º e 27º, sendo 0 o valor quando este se encontra paralelo ao horizonte. Este ângulo tem de ser ajustado por forma a que o prato rotativo esteja totalmente posicionado dentro do campo de visão do sensor *Microsoft Kinect*. É importante encontrar um ângulo em que o objeto tenha a máxima visibilidade pelo sensor e certificar que todos os lados do objeto possam ser vistos pelo sensor quando o objeto é rodado. Contudo, nesta etapa o prato rotativo e o objeto ainda não estão inteiramente isolados do resto da cena. Para tal, vai se definir uma caixa imaginária que delimita a área de interesse, sendo que qualquer ponto fora desses parâmetros será desprezado. A nuvem de pontos resultante irá conter essencialmente a informação *3D* referente ao objeto e à plataforma rotativa desprezando grande parte da informação da cena envolvente. Por último, será aplicado uma técnica de segmentação de planos para excluir a informação tridimensional da plataforma.

4.3.4 Filtro Passthrough

As nuvens de pontos adquiridas pelo sensor *Microsoft Kinect* têm por volta de 30.000 pontos. Nuvens de pontos com esta dimensão requerem tempos de computação bastante elevados. A informação desnecessária para além de aumentar a probabilidade de ocorrerem erros, também garante que o tempo para comparar todas as características das nuvens de pontos seja muito superior. Portanto, torna-se indispensável excluir os detalhes que não sejam relevantes. Este algoritmo permite eliminar muitos pontos circundantes, bem como reduzir algum do ruído presente na cena. Como o objetivo desta dissertação é capturar pequenos objetos localizados numa superfície plana, o fundo ruidoso pode ser descartado com este filtro.

Nesta dissertação, foi implementado um filtro *Passthrough* para cada uma das dimensões (*x, y, z*) definindo um limite máximo e mínimo para a janela de visualização da nuvem de pontos capturada. Isto é, a partir do momento que o filtro tenha sido aplicado, todos os pontos resultantes pertencem $n_k \in [n_{min}, n_{máx}]$ metros, sendo n_k o eixo de coordenadas do sensor *Kinect*. A interseção destes três intervalos define a região de interesse no qual irá ser colocado o objeto que será digitalizado. A *Figura 47* ilustra a informação tridimensional de um objeto incluído nessa região de interesse.



(a) **(b)**

Figura 47: Nuvem de pontos inicial (a) Nuvem de pontos após aplicação do filtro (b)

4.3.5 Segmentação dos objetos

Após ter sido implementado o filtro *Passthrough*, a próxima etapa consiste em aplicar um método de segmentação do plano que contém o objeto a digitalizar. A segmentação, como o próprio nome sugere é o processo de divisão de uma imagem em regiões que posteriormente podem ser separadas por forma a serem analisadas separadamente. Esta técnica permite isolar os pontos do objeto em questão dos pontos referentes à plataforma rotativa.

Para a segmentação formas geométricas é criado um objeto *SACSegmentation* e necessário definir o tipo de modelo e método. Para além disso, uma distância padrão tem de ser especificada de modo a que seja possível determinar a proximidade de um ponto em relação ao modelo paramétrico por forma a ser considerado *inlier*.

Nesta dissertação, o método utilizado foi o *RANSAC* porque é a escolha mais robusta e simples para estimar os parâmetros do modelo desejado. Assim sendo, este algoritmo vai procurar quais os pontos da nuvem de pontos que se encaixa na equação do modelo escolhido, devolvendo os parâmetros desse modelo. Após ter sido atingido o número máximo de iterações, a equação que tiver o número máximo de *inliers* é considerada equação final do plano.

Alguns dos modelos implementados na biblioteca *PCL* incluem: linhas, planos, cilindros e esferas. O modelo que foi implementado neste projeto envolve a deteção de planos paralelos ao sensor (*SACMODEL_PARALLEL_PLANE*). Este modelo permite a deteção de superfícies planas, tais como mesas, chão, paredes presentes numa determinada cena. Neste caso, o objetivo deste algoritmo é a deteção do plano que suporta o objeto em análise.

Após a segmentação do plano é necessário extrair o *cluster* que contém o objeto. A extração euclidiana é o método de segmentação mais simples de todos. Este método permite a procura e a segmentação de objetos individuais em diversos *clusters* que se encontram num determinado plano. Para a extração dos *clusters* é criado um objeto *EuclideanClusterExtraction* do tipo *PointXYZRGBA*, que é o tipo da nuvem de pontos obtida pelo sensor. Uma estrutura *kd-tree* é utilizada como método de procura neste algoritmo de extração. Um vetor de índices é criado para armazenar os índices de cada um dos *clusters* detetados. Em que o índice 0 corresponde ao primeiro *cluster* da nuvem de pontos, e simultaneamente, corresponde ao *cluster* com maior número de pontos. O parâmetro mais sensível é a tolerância do *cluster*. Se o valor pré definido for muito pequeno, um único objeto pode ser visto em diversos *clusters*, caso contrário poderá acontecer que múltiplos objetos podem ser vistos num único *cluster*. Os resultados finais da extração dependem muito do tamanho máximo e mínimo de pontos por *cluster*. Após os parâmetros terem sido definidos, os *clusters* podem ser extraídos da nuvem de pontos e guardados no vetor de índices. Para separar cada *cluster* do vetor tem de se criar uma nuvem de pontos para cada entrada e escrever todos os pontos do cluster atual nessa nuvem de pontos. A *Figura 48* ilustra a segmentação do plano e a identificação do *cluster* que contém o objeto.

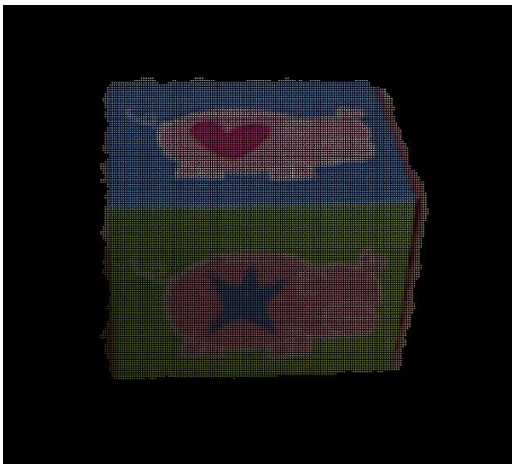


Figura 48: Nuvem de pontos após segmentação do plano

4.4 Pré-processamento

O módulo de pré-processamento contém diversos algoritmos utilizados com o objetivo de melhorar a qualidade e tamanho da informação contida na nuvem de pontos. O ruído presente nas nuvens de pontos tem uma forte dependência nas características resultantes. Devido à necessidade de se aplicar técnicas de descrição de características locais, a implementação de filtros permite reduzir a influência do ruído nas medições obtidas. Para além disso, em algumas aplicações a redução da resolução da nuvem de pontos é suficiente para uma boa deteção de características, o que traduz uma melhoria no tempo de processamento.

4.4.1 Remoção de NAN's

A nuvem de pontos obtida a partir do sensor pode conter diversos erros de medição e/ou imprecisões. Um dos problemas pode ser caracterizado pela existência de valores nas nuvens de pontos que não representam um número válido, denominado *NaN* (*Not a Number*). A presença de *NaN's* indica que ocorreu um problema na medição da distância para um determinado ponto, sendo caraterizado pela elevada ou baixa distância em relação ao sensor ou pela influência das propriedades da superfície de um determinado objeto. Muitos dos algoritmos da biblioteca *PCL* podem falhar na presença destes valores não válidos. Para prevenir este problema, a solução passa por remover todos estes valores antes de se efetuar o processamento da nuvem de pontos. O módulo *filters* da biblioteca *PCL* contém uma classe especifica (*pcl::removeNaNFromPointCloud*) para remover estes valores da nuvem de pontos.

4.4.2 Matriz de transformação

As nuvens de pontos podem ser manipuladas através de uma determinada transformação, que pode envolver uma translação, rotação, entre outras. Como mencionado anteriormente, o sistema de coordenadas das nuvens de pontos obtidas pelo sensor é diferente do *framework OpenNI*. Para tal, é utilizada uma matriz de transformação *4x4* para definir um sistema de coordenadas personalizado e fazer os ajustes necessários para uma melhor visualização do objeto. Nesta dissertação, foi aplicada uma transformação aos eixos coordenados *z* e *y*. A matriz de transformação no eixo *z* e no eixo *y* é dada pelas seguintes equações:

$$R_Z = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.1)$$

$$R_Y = \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.2)$$

4.4.3 Redução da resolução e remoção dos outliers

Para reduzir o tempo de processamento é necessário diminuir a resolução de determinada nuvem de pontos (*downsampling*). Embora as nuvens de pontos já tenham sido filtradas inicialmente recorrendo ao filtro *Passthrough*, após a concatenação das duas nuvens de pontos existe demasiada informação para se processar. Através da utilização de uma técnica de decomposição espacial, denominada voxelização, é possível diminuir a complexidade da nuvem de pontos. Obtém-se uma nuvem de pontos equivalente ou capaz de representar à nuvem de pontos original apesar de ter menos pontos. Desta forma, é possível definir a distância mínima (*leaf-size*) entre pontos para a precisão desejada. Neste caso, o tamanho de cada *voxel* foi definido como um cubo com 1 mm de lado, isto é, apenas existirá um ponto por cada milímetro cúbico (*Figura 49*). Na *Tabela 8* pode-se observar que quanto maior for o valor da distância, menor é o número de pontos à saída do filtro.

Tabela 8: Redução da resolução de uma nuvem de pontos

Distância mínima (mm)	**S/filtro**	**1.0**	**2.0**	**3.0**	**4.0**	**5.0**
Nº pontos	14939	14939	9104	5116	3186	2177

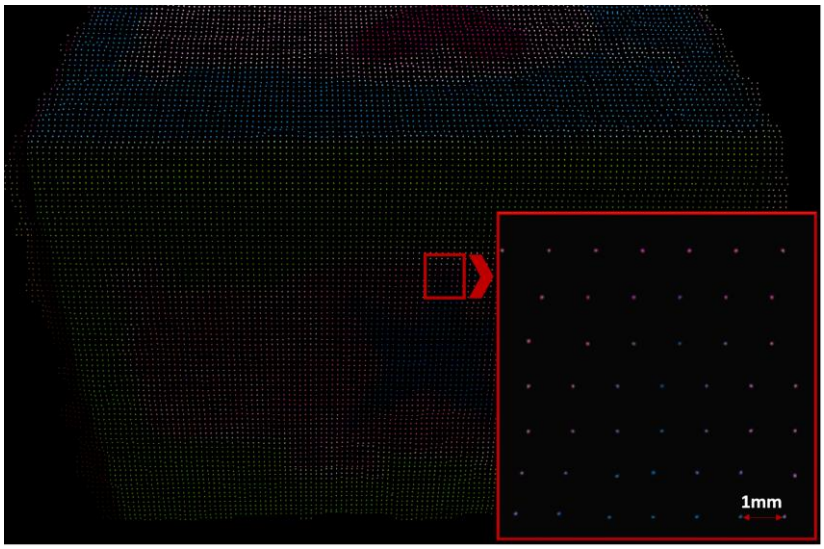


Figura 49: Nuvem de pontos com resolução de 1 mm

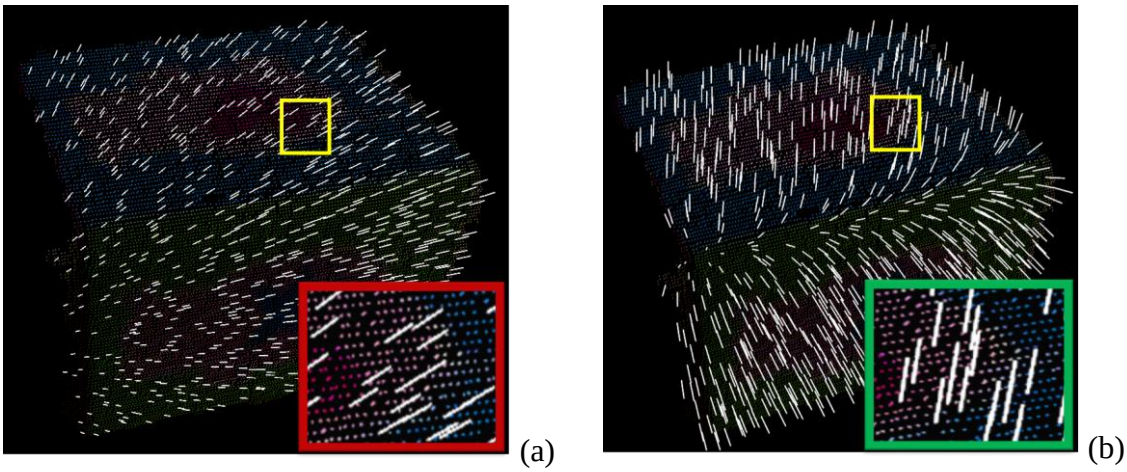
Para além do *downsampling* da nuvem de pontos foi também implementado um algoritmo de remoção de outliers baseado no raio da vizinhança (*Radius Outlier Removal*). As nuvens de pontos adquiridas a partir do sensor *Kinect* podem conter erros de medida levando ao aparecimento de *outliers*. Este método têm como objetivo a remoção desses pontos que possam originar erros na estimativa local das características de uma nuvem de pontos. Através da análise da vizinhança de um determinado ponto, se dentro de um raio *r* pré-definido pelo utilizador, existir um número mínimo de pontos então o ponto que está a ser analisado é considerado *inlier*. Neste caso, cada ponto tem de ter no mínimo 15 vizinhos num raio de *2 cm*, caso contrário esse ponto (*outlier*) será removido da nuvem de pontos.

4.4.4 Estimativa das normais

Por último, após algumas etapas com o objetivo de filtrar a nuvem de pontos adquirida pelo sensor, é calculada as normais da superfície. A normal da superfície de um ponto é uma das características mais importantes para diferenciar diversos pontos de uma nuvem de pontos. As coordenadas de um ponto não é uma característica suficiente, pois quando se tenta interpolar a informação correspondente de duas nuvens de pontos esta característica torna-se inútil. Para tal utiliza-se a informação das normais, pois estas terão valores semelhantes quando calculadas para o mesmo local da superfície em ambas as nuvens de pontos. Esta informação é utilizada durante a fase de processamento para deteção dos descritores das características.

Como explicado anteriormente, uma normal da superfície num ponto deve ser estimado a partir da vizinhança circundante desse ponto. Sem entrar em muito detalhe, basta assumir que

a escala para determinação da vizinhança de um ponto tem que ser selecionado com base no nível de detalhe requerido pela aplicação. Ou seja, se a curvatura na extremidade de um determinado objeto é importante, o fator de escala deve ser pequeno o suficiente para capturar essa característica, e grande caso se pretenda o contrário. Assim sendo, a utilização de uma grande escala pode distorcer as normais e, possivelmente, suprimir detalhes finos, esse comportamento pode ser visto na *Figura 50*.



(a) (b)

Figura 50: Estimativa das normais com fator de escala de 1.0 (a) e de 0.05 (b)

A biblioteca *PCL* disponibiliza uma implementação adicional para estimativa das normais da superfície que utilizam paradigmas *multi-core/multi-threaded* a partir do modelo *OpenMP* para acelerar o cálculo destas características. Como tal, esta implementação foi utilizada neste projeto de investigação.

4.5 Processamento

Quando se processa a informação tridimensional de uma nuvem de pontos, os descritores de características (*features*) são extraídos de um pequeno conjunto de pontos, denominados pontos de interesse (*keypoints*). O custo computacional dos descritores é demasiado elevado, por isso não faz sentido extrair as características de todos os pontos que pertencem à nuvem de pontos. O propósito dos detetores de pontos de interesse é determinar os pontos que são diferentes por forma a permitir uma eficiente descrição das características geométricas e suas correspondências em relação a vários pontos de vista.

4.5.1 Pontos de interesse

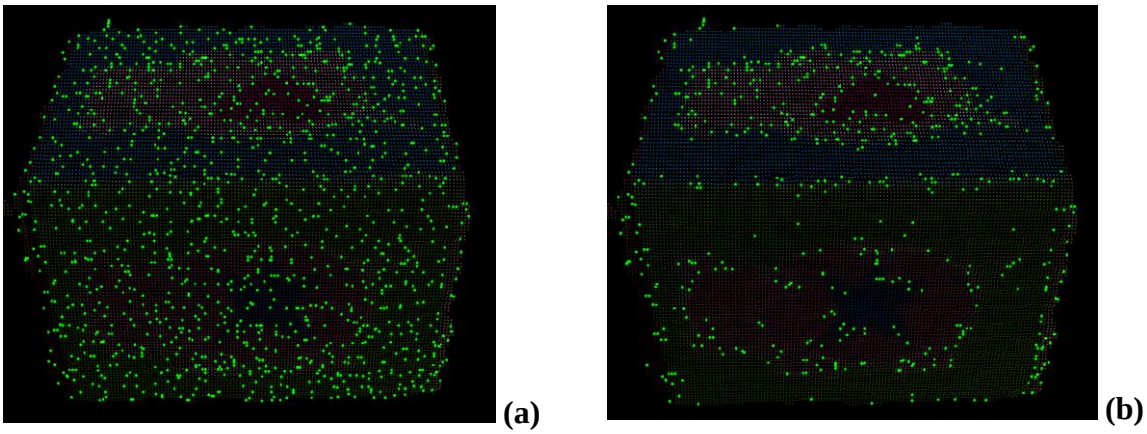
Os pontos de interesse são pontos especiais capazes de serem encontrados numa nuvem de pontos similar. Estes pontos são geralmente posicionados nos contornos de formas e em locais onde o gradiente de cor/brilho varia rapidamente. A biblioteca *PCL* oferece dois algoritmos de deteção de pontos de interesse: o *SIFT* e o *Harris Keypoints* para ajudar na procura destes pontos na nuvem de pontos, mas eles requerem a informação da imagem *RGB*. Neste projeto de investigação é utilizado o método *SIFT* de forma a encontrar os pontos de interesse de duas nuvens de pontos e obter-se a transformação rígida entre elas. Os pontos de interesse adquiridos a partir deste algoritmo são pontos altamente descritivos numa imagem *RGB*, sendo obtidos a partir da comparação de um único píxel com os seus vizinhos. Para tal, os valores da dimensão *RGB* da nuvem de pontos são convertidas para valores de intensidade.

A implementação deste algoritmo tem como base 4 parâmetros: escala mínima, número de oitavas, número de escalas por oitava e o contraste mínimo. A escala mínima no espaço *3D* é dado pela distância mínima entre pontos numa nuvem de pontos. O número de oitavas e de escalas por oitava são recomendados por *David Lowe* [110] para serem definidos por 4 e 5, respetivamente. A escolha dos valores dos parâmetros afeta a resposta do algoritmo e, por sua vez a deteção local dos descritores de características que diretamente influencia o processo de registo das nuvens de pontos. Na *Tabela 9* pode ser observado que quanto maior for o valor do parâmetro referente ao contraste mínimo menor será o número de pontos de interesse extraídos da nuvem de pontos.

Tabela 9: Valores de contraste mínimo para método *SIFT*

Contraste mínimo	**0.05**	**0.1**	**0.2**	**0.5**	**0.7**	**1.0**	**1.5**	**2.0**	**3.5**
Nº pontos de interesse	1507	1503	1431	1055	888	668	441	313	84

No nosso conjunto de dados o valor mais adequado para o contraste mínimo é 1.0. Os valores de contraste mínimo inferiores a este são capazes de gerar diversos pontos de interesse no meio de nenhum caraterística geométrica relevante. A *Figura 51* ilustra a comparação feita em dois valores distintos de contraste mínimo na mesma nuvem de pontos.



(a) **(b)**

Figura 51: Deteção de pontos de interesse com contraste mínimo de 0.1 (a) e de 1.0 (b)

4.5.2 Descritores de características

Os descritores de características são utilizados para encontrar as correspondências entre pontos para posteriormente ser aplicado no registo de nuvem de pontos. Estes descritores apresentam invariância em relação à posição que representam na superfície subjacente de um ponto dentro de um raio de busca definido pelo utilizador. A relação entre cada ponto nesta superfície é baseada no valor dos três ângulos entre as normais e na distância euclidiana entre os pontos. O desempenho de qualquer descritor de características depende da robustez das normais da superfície estimadas. Quanto menor for o número de caraterísticas e melhor elas caraterizarem os dados, mais eficientes serão os passos subsequentes de interpretação de dados.

O algoritmo implementado nesta dissertação foi *FPFH*, pois o desempenho e tempo de cálculo destas características é superior ao algoritmo *PFH*. Este algoritmo depende da nuvem de pontos de entrada (neste caso, os pontos de interesse), da superfície que será utilizada para a procura dos vizinhos mais próximos (ou seja, a nuvem de pontos original), da estimativa das normais previamente calculadas e de um raio que será estabelecido para a determinação do conjunto de pontos vizinhos sobre o qual se irá estimar os descritores das características locais. No entanto, a escolha de um conjunto de pontos da vizinhança não é fácil, e baseia-se em duas dependências: o número de vizinhos k e a densidade da nuvem de pontos φ. Este problema está relacionado com a discussão presente na *secção 4.4.4* acerca da escala das normais da superfície. Para se evitar erros na etapa de processamento das nuvens de pontos, tem de se garantir que o raio utilizado para deteção dos descritores das características é superior ao raio utilizado para estimar as normais da superfície. Para tal, como regra informal, a escala da estimativa das normais é definida como pelo menos metade do raio do descritor das características da superfície do objeto.

A biblioteca *PCL* disponibiliza uma implementação otimizada do algoritmo *FPFH* que utiliza paradigmas *multi-core/multi-threaded* através da biblioteca *OpenMP* (a partir da classe *pcl::FPFHEstimationOMP)*. Esta classe é utilizada para acelerar o processo de cálculo dos descritores de características, o que traduz um melhor desempenho global por um fator de número de processadores da máquina.

4.5.3 Correspondências

Após terem sido calculados os descritores de características para os diversos pontos de interesse é necessário encontrar as correspondências entre esses pontos nas duas nuvens de pontos. Uma correspondência pode ser definida simplesmente por um par de características que possuem valores de correspondência semelhantes. A partir das melhores correspondências obtidas será possível estimar a transformação rígida entre as duas nuvens de pontos. Contudo, a introdução de informação falsa e dados incorretos pode comprometer o processo geral de registo das nuvens de pontos.

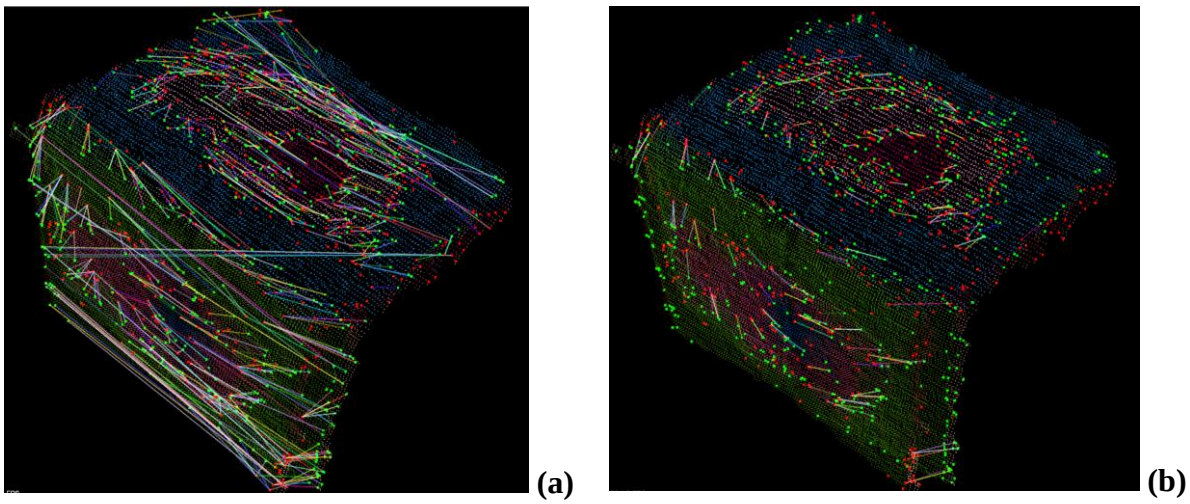
A biblioteca *PCL* possui a classe *pcl::registration::CorrespondenceEstimation* capaz de procurar este tipo de informação, a partir ou das coordenadas dos pontos (*point matching*) ou do conjunto dos descritores de características (*feature matching*) das duas nuvens de pontos.
Para além disso, existem dois tipos de estimativa de correspondências: um baseado na estimativa direta, ou seja, a procura de correspondências na nuvem de pontos B para todos os pontos da nuvem de pontos A, e o outro baseado na estimativa reciproca de correspondências, isto é, a procura de correspondências entre a nuvem A para a B, e a partir da nuvem B para a A, sendo utilizado apenas a interseção. Nesta dissertação, este método foi implementado por forma a detetar as correspondências com base unicamente nos descritores de características.

4.5.4 Rejeição de más correspondências

Após terem sido encontradas as correspondências entre nuvens de pontos é possível rejeitar algumas delas baseadas em condições específicas. A biblioteca *PCL* disponibiliza uma classe denominada *pcl::CorrespondenceRejectorSampleConsensus* capaz de rejeitar as más correspondências desse conjunto. Este algoritmo de rejeição de correspondências é baseado no método *RANSAC*, sendo utilizado para estimar uma transformação para um conjunto de correspondências e eliminar as correspondências de *outliers* com base na distância euclidiana entre os pontos após a melhor transformação ter sido aplicada à nuvem de pontos de origem. Este método é muito eficaz em manter o algoritmo *ICP* convergindo para um mínimo local,

sendo capaz de produzir correspondências ligeiramente diferentes. Além disso, ele fornece uma boa estimativa inicial para o algoritmo *ICP*. Através da configuração dos parâmetros deste algoritmo é possível controlar a quantidade de correspondências resultantes. A função *setInlierTreshold* é utilizada para definir a distância máxima entre pontos correspondentes. Por fim, através da função *setMaxIterations* é possível estabelecer o número máximo de iterações para o algoritmo completar o processo.

Na *Figura 52* é possível visualizar a estimativa das correspondências entre dois conjuntos de dados, sendo utilizadas para o efeito duas nuvens de pontos com um pequeno desfasamento entre si. A *Figura 52-a* ilustra todas as correspondências encontradas entre dois conjuntos de dados ligeiramente desfasados entre si. Na *Figura 52-b* é possível visualizar as correspondências encontradas após ter sido aplicado o algoritmo de rejeição de correspondências baseado no método *RANSAC*.



(a) **(b)**

Figura 52: Estimativa das correspondências (a) e correspondências obtidas após rejeição (b)

4.6 Registo de nuvens de pontos

O processo de registo consiste no alinhamento de várias nuvens de pontos num único modelo capaz de descrever de forma completa um determinado objeto. A partir das correspondências obtidas através dos descritores de características dos pontos de interesse de cada nuvem de pontos é possível encontrar um método capaz de alinhar corretamente estas duas nuvens de pontos. Tal como mencionado anteriormente, o algoritmo *ICP* implementado na biblioteca *PCL* tem as suas limitações na interseção de duas nuvens de pontos registadas. Como tal, este algoritmo será utilizado para aperfeiçoar o alinhamento após uma transformação inicial

ter sido executada nesse conjunto. Assim sendo, o processo de registo de nuvens de pontos é dividido em 2 partes: o alinhamento inicial e o alinhamento refinado. A *Figura 53* ilustra as etapas de processamento que antecedem o processo de registo de nuvens de pontos.

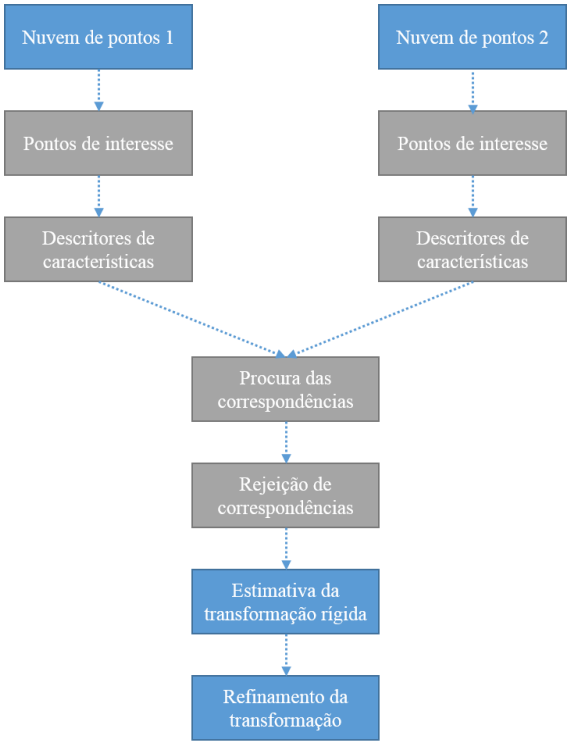


Figura 53: Processo de registo de duas nuvens de pontos

O processo de registo pretende resolver a transformação e rotação de uma nuvem de pontos capturada para se encaixar no modelo das nuvens de pontos previamente capturadas. A concatenação de duas nuvens de pontos resulta num modelo parcialmente construído do modelo global que se pretende de um objeto. Na primeira iteração do processo de registo de duas nuvens de pontos, o modelo global está vazio e, como tal a nuvem de pontos resultante é movida para o modelo global. Na iteração seguinte a nuvem de pontos que se segue, capturada a partir de um ângulo ligeiramente diferente, será processada com o modelo global obtido anteriormente, e assim sucessivamente. Na última iteração, é feita uma comparação entre o último modelo obtido e a primeira nuvem de pontos capturada por forma a tornar este processo um ciclo fechado sem qualquer perda de informação. A *Figura 54* ilustra o processo de emparelhamento de nuvens de pontos implementado.

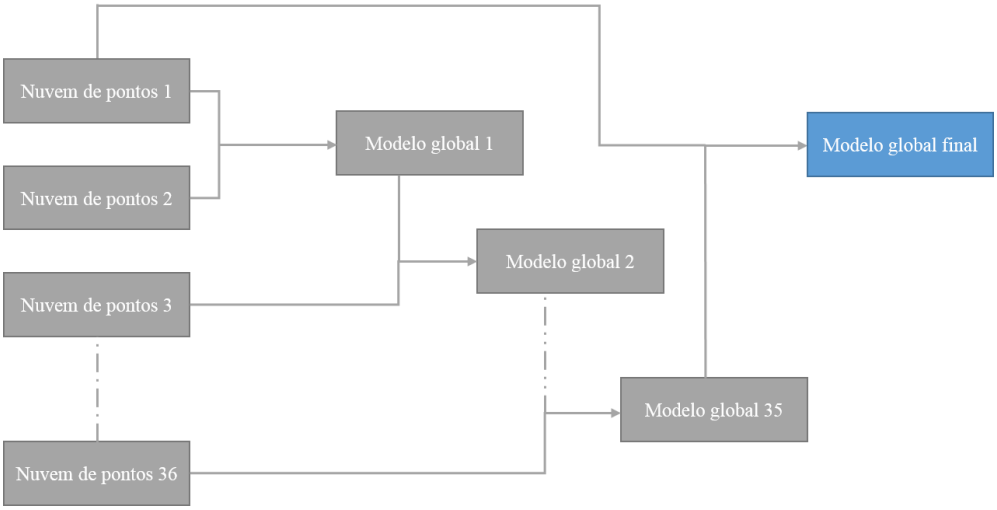


Figura 54: Processo de emparelhamento de nuvens de pontos

4.6.1 Alinhamento Inicial

A etapa de estimativa de transformação inicial é realizada após a correspondência dos pontos entre as nuvens de pontos de origem (*source*) e de destino (*target*) terem sido filtradas. O objetivo do alinhamento inicial consiste em encontrar a transformação para a nuvem de pontos capturada em relação ao modelo global previamente processado. Assim sendo, dependendo da qualidade das correspondências encontradas, as áreas que se intersetam das duas nuvens de pontos são quase coincidentes após esta transformação, portanto esta etapa pode servir como boa estimativa inicial para o alinhamento de refinamento *ICP*.

Na biblioteca *PCL*, existem vários métodos de transformação de nuvens de pontos sem a necessidade de qualquer parâmetro. Nesta dissertação, dois métodos foram implementados: *TransformationEstimationSVD* e *TransformationEstimationLM*. O primeiro método é uma solução em forma fechada com base na decomposição do valor singular de uma matriz de covariância de dados, proporcionando a melhor solução possível para um único passo. O segundo método é um algoritmo iterativo baseado em mínimos quadrados.

4.6.2 Alinhamento Refinado

O alinhamento inicial é uma transformação rígida que pode não ser totalmente aceitável para uma análise de deformação final. Apesar disso, um algoritmo de refinamento é utilizado para encontrar uma transformação mais precisa entre as duas nuvens de pontos. Como mencionado anteriormente, o método *ICP* é o algoritmo mais popular aplicado nesta área. Este algoritmo é utilizado para alinhar as duas nuvens de pontos o mais próximo possível. Na

biblioteca *PCL* existem diversas funções aplicadas para definir os parâmetros de controlo deste processo. A função *setMaximumIterations* é utilizada para controlar o número máximo de iterações de execução deste algoritmo. Para além desta, a função *setTransformationEpsilon* é importante para definir a diferença máxima entre duas nuvens de pontos consecutivas. Estes dois valores são utilizados para determinar o momento em que o método *ICP* alcança a convergência. Através do parâmetro da função *setRANSACOutlierRejectionThreshold* é possível definir o limite de distância para cada *inlier* para o ciclo interno de rejeição de *outlier* baseado no método *RANSAC.* O método considera um ponto *inlier*, se a distância entre o índice de dados da nuvem de pontos alvo e da nuvem de pontos de origem transformada for menor que um certo limite de distância pré-definido. Por fim, o parâmetro da função *setMaxCorrespondenceDistance* permite definir o limite máximo de distância entre dois pontos correspondente entre a nuvem de pontos de origem e alvo. Se a distância for maior que esse limite, os pontos são ignorados no processo de alinhamento.

4.7 Processamento e reconstrução do modelo

Após o processo de registo de nuvens de pontos ter sido completo ainda é presente a existência de erros e de ruído na superfície do modelo resultante. Esta dificuldade leva à utilização de técnicas de processamento de nuvens de pontos capazes de suavizar e aumentar a resolução da informação *3D* obtida no final desta operação. O método implementado para este efeito é baseado no algoritmo *Moving Least Squares* (*MLS*). As superfícies do objeto são então reconstruídas a partir da nuvem de pontos resultante deste algoritmo. Como tal, o algoritmo *Marching Cubes* (*MC*) e o *Greedy Triangulation* (*GT*) são algumas das técnicas implementadas na biblioteca *PCL* para a reconstrução da superfície de determinado objeto. Estes algoritmos dizem respeito a formulações de reconstrução das nuvens de pontos em malhas tridimensionais, vulgarmente denominado *mesh.* Assim sendo, a partir desta malha a nuvem de pontos pode ser exportada como formato de ficheiro *.ply* ou *.stl*. Estes formatos de ficheiro contêm os dados poligonais de um objeto e podem ser utilizados em várias aplicações, como programas de *CAD*.

4.7.1 Moving Least Squares

O *Moving Least Squares* (*MLS*) é um algoritmo de suavização dos dados contidos nas nuvens de pontos. Ele é utilizado para aproximar a superfície original de um objeto a partir da vizinhança local dos diversos pontos. A ideia básica consiste em primeiro encontrar o plano de

referência local e depois aplicar uma aproximação através de uma função polinomial no conjunto de dados. Este algoritmo pertence a uma classe de métodos de interpolação de malhas tridimensionais, e existem várias formas competitivas para implementar este método. Todos estes métodos têm um objetivo em comum que consiste em diminuir o ruído dos dados presentes na superfície. Nesta dissertação, dois métodos foram utilizados: o primeiro é baseado na dilatação de uma rede *voxel*, denominado *VOXEL_GRID_DILATATION*, e o segundo é baseado na dilatação do plano local de cada ponto, intitulado *SAMPLE_LOCAL_PLANE*.

De uma maneira geral, para o primeiro método é aplicada uma rede *voxel* à nuvem de pontos de entrada, com um tamanho pré-definido, e de seguida será dilatado durante diversas iterações, sendo os pontos resultantes projetados na superfície do ponto mais próximo da nuvem de entrada. O resultado deste método é uma nuvem de pontos com os buracos preenchidos e com uma densidade constante. Para tal, as funções *setDilationVoxelSize* e *setDilationIterations* permitem definir o tamanhão do *voxel* e o número de etapas de dilatação para a rede voxel, respetivamente. No segundo método, para cada ponto é dilatado o plano local através da criação de pontos num círculo de raio fixo e tamanho da passo fixo. De seguida, a partir do polinómio utilizado é calculada a normal nessa posição e adicionado o deslocamento ao longo da normal. Para rejeitar os pontos ruidosos, é aumentado o valor do parâmetro para o número de pontos necessário a fim de se estimar o ajuste polinomial local. Para isso, as funções *setUpsamplingRadius* e *setUpsamplingStepSize* permitem estabelecer o raio do círculo e o tamanho do passo no plano local do ponto que será amostrado.

Por outro lado, para a configuração do algoritmo é necessário estabelecer os parâmetros das várias funções implementadas. O parâmetro da função s*etSearchRadius* define o raio da esfera que será utilizada para a determinação dos pontos vizinhos. A função *setSqrGaussParam* é utilizada para estabelecer o parâmetro utilizado para a ponderação da distância com base na vizinhança, sendo o quadrado do raio de busca o valor recomendável na maioria dos casos. Por último, a função *setPolynomialFit* define se a superfície e normais são aproximadas aplicando um polinómio, ou apenas via estimativa da tangente. A função *setPolynomialOrder* é utilizada para definir a ordem do polinómio que será aplicado neste processo.

4.7.2 Greedy Triangulation

Nesta dissertação, o algoritmo *Greedy Triangulation* (*GT*) foi implementado para a reconstrução dos objetos digitalizados. Este algoritmo de triangulação da superfície utiliza a informação presente na nuvem de pontos e das normais da superfície para criar uma malha de

triângulos com base em projeções da vizinhança local. O método funciona através da manutenção de um conjunto de pontos a partir do qual a malha pode ser originada e capaz de se estender até que todos os pontos estejam conectados. Pode ser aplicado a nuvens de pontos desorganizadas, a partir de uma ou várias capturas, e ter múltiplas partes conectadas. O resultado é melhor se a superfície de determinado objeto for suavizada localmente e existirem transições suaves entre regiões com diferentes densidades de pontos. A triangulação é calculada localmente, através da vizinhança local de um ponto ao longo das normais dos diversos pontos, e conectando pontos desconectados.

A configuração deste algoritmo depende dos parâmetros aplicados nas diversas funções implementadas. Assim sendo, a função *setMaximumNearestNeighbors* e *setMu* estabelecem o tamanho dos pontos da vizinhança. A primeira define o número de pontos que são pesquisados e a segunda indica a distância máxima aceitável para um ponto ser considerado em relação à distância do ponto mais próximo, por forma a ajustar as mudanças de densidade nas diversas regiões da superfície do objeto. O parâmetro da função *setSearchRadius* define o comprimento máximo das extremidades para cada triângulo. As funções *setMinimumAngle* e *setMaximumAngle* são utilizadas para indicar os ângulos mínimos e máximos em cada triângulo. Por fim, as funções *setMaximumSurfaceAngle* e *setNormalConsistency* designam o ângulo máximo da superfície e a *flag* para manter a orientação das normais de forma consistente. Estas duas funções são destinadas a lidar com as situações em que existam extremidades acentuadas, onde os dois lados da superfície estão muito próximos um do outro. Desta forma, os pontos não serão conectados ao ponto atual, se as suas normais se desviarem do ângulo especificado.

5. PLATAFORMA ROTATIVA

No presente capítulo é apresentado o sistema que serve de apoio à aquisição dos dados a partir do sensor Kinect. O estudo desenvolvido no sentido de construir esta estrutura, assim como todo o hardware e software utilizado para a implementação da plataforma rotativa é descrito nesta secção.

5.1 Introdução

Para criar um modelo *3D* completo que inclua todas as partes constituintes de um objeto, o *scanner 3D* deve primeiro adquirir várias capturas tridimensionais desse objeto a partir de diferentes direções. Assim que todas as partes tenham sido capturadas, o próximo passo consiste em juntar todas as capturas obtidas de forma a criar um modelo tridimensional. O processo de aquisição de capturas *3D*, e posteriormente a sua fusão e alinhamento requer tempo e precisão. Para facilitar o método de captura, a solução passa pela utilização de uma plataforma rotativa. A plataforma permite automatizar o processo de aquisição de múltiplas capturas de um objeto, sendo possível saber em qualquer etapa a orientação de cada delas em relação à captura inicial.

5.2 Estrutura da plataforma

Para o desenvolvimento de uma plataforma rotativa são indispensáveis um conjunto de materiais, e para tal é necessária uma estrutura capaz de suportar todos estes componentes e ainda o objeto que se deseja digitalizar. O objeto será colocado sobre esta estrutura na qual a plataforma vai rodando num determinado ângulo de forma a que este possa ser visto para cada um dos pontos de vista. A plataforma construída permite rodar o objeto *360º*, sabendo a cada momento o ângulo preciso em que este se encontra para que seja possível obter as coordenadas globais dos pontos digitalizados. A finalidade de construir um tal dispositivo é o de ter uma plataforma que possa manter um peso razoável, sem entrar em colapso e que possa rodar livremente usando um motor de passo, garantindo uma maior precisão de seu dispositivo físico. Quanto maior for a precisão do dispositivo, maior será a consistência das nuvens de pontos quando alinhadas em conjunto. Nesta dissertação, foi realizado um projeto em *SOLIDWORKS* da estrutura capaz de suportar os diversos componentes e objeto que se pretende digitalizar, sendo capaz de garantir a credibilidade da plataforma rotativa (*Figura 55*).

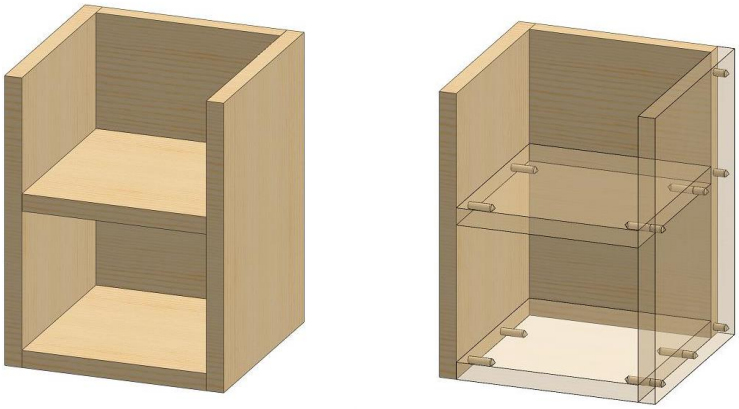


Figura 55: Projeto *SOLIDWORKS* da estrutura da plataforma rotativa (*Anexo I*)

5.3 Material

Foi realizado um estudo prévio para determinar quais os materiais que deviam ser utilizados para implementar a plataforma rotativa. Este projeto foi construído a partir da utilização de componentes baratos e outros materiais disponíveis nas oficinas do *Departamento de Eletrónica Industrial* e no *Laboratório de Automação e Robótica* (*LAR*). Para a construção da plataforma foram usados os seguintes materiais:

- *1 Arduino*
- *1 Acoplador mecânico*
- *1 Base circular*
- *1 Controlador*
- *1 Fonte de alimentação*
- *1 Motor de passo*

5.3.1 Motor de passo

Um motor de passo converte pulsos eletrónicos em movimento mecânico proporcional. Cada rotação do eixo do motor é composta por uma série de passos individuais discretos. Um passo é definido como o ângulo de rotação produzido pelo eixo de saída cada vez que o motor recebe um impulso de passo. Cada passo faz com que o eixo rode um certo número de graus, sendo este passo dependente do arranjo dos dentes do motor. A principal vantagem dos motores de passo é o facto de estes serem capazes de controlar a posição exata sem a necessidade de retorno da posição (*feedback*).

O motor de passo utilizado na plataforma rotativa foi desenvolvido pela empresa *DFRobot (Figura 56)*. É um motor híbrido de 2 fases, aplicado em máquinas *CNC* e impressoras *3D* [134]. As principais especificações deste motor são apresentadas na *Tabela 10*.

Tabela 10: Principais especificações do motor de passo [134]

Motor de Passo (Modelo 42BYGH40-170-4A)	
Ângulo de passo (graus)	1.8
Tensão nominal	12 V
Corrente nominal	0.33 A
Binário	3.5 Kg*cm

Figura 56: Motor de Passo

5.3.2 Big Easy Driver

Na plataforma rotativa, a direção e ângulo do motor de passo é controlado pela placa *Big Easy Driver* (*Figura 57*). A *Big Easy Driver* é uma placa de controlo de motores de passo bipolares até *2A* por fase. Para além disso, esta placa suporta motores de passo com tensões de funcionamento até *35V* e possui um regulador de tensão (*5V/3.3V*) na própria placa [135].

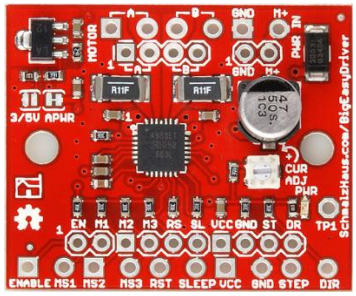


Figura 57: Controlador *Big Easy Driver* [135]

5.3.3 Arduino Mega 2560

O *Arduino Mega 2560* é uma placa de desenvolvimento baseada no microcontrolador *ATmega2560*. Este dispositivo contém 54 pinos digitais de entrada/saída, 16 entradas analógicas, 4 *UARTs* (portas séries), um cristal oscilador de *16 MHz*, uma conexão *USB*, um *jack* de alimentação, um *header ICSP* e um botão *reset*. Ele contém tudo o que é necessário para programar o microcontrolador, basta liga-lo a um computador com cabo *USB* ou adaptador *AC-DC* para alimentar a placa [136]. Na plataforma rotativa, esta placa de desenvolvimento será utilizada para configurar os sinais de controlo da direção e do ângulo de passo que serão transmitidos para a placa controladora referida anteriormente.

Figura 58: Arduino Mega 2560 [136]

5.4 Controlo do motor

O software de controlo do motor de passo foi desenvolvido em linguagem *C* utilizando a plataforma de desenvolvimento *Atmel Studio 6.2* por forma a interagir com o *Arduino Mega* para configurar o movimento do motor de passo. O *Atmel Studio* [137] é uma plataforma de desenvolvimento integrado para a implementação e depuração de aplicações baseadas em microcontroladores *Atmel AVR*. Este software oferece um ambiente simples para a elaboração de diversas aplicações em linguagem *C/C++* ou código *assembly*.

Neste projeto de investigação, uma simples configuração é realizada por forma a interligar o *Arduino*, o controlador e o motor de passo. Os quatro fios do motor de passo são conectados à placa controladora *Big Easy Driver* (*A* e *B*), tendo em atenção as ligações dos enrolamentos das bobinas internas do motor. Esta placa controladora é alimentada externamente por uma fonte de alimentação de *12V*. Os pinos *STEP* e *DIR* desta placa são

conectados aos pinos 13 e 12 do *Arduino Mega*, respetivamente. A *Figura 59* ilustra as ligações feitas entre estes 3 componentes.

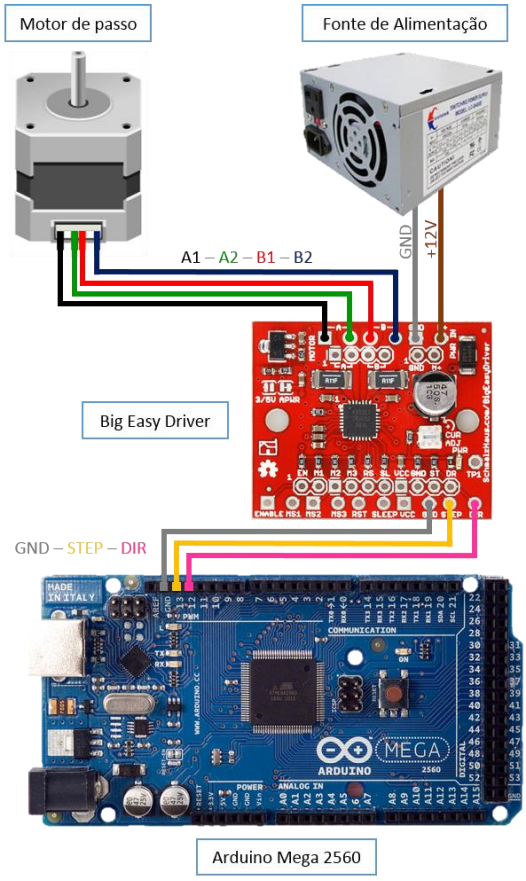
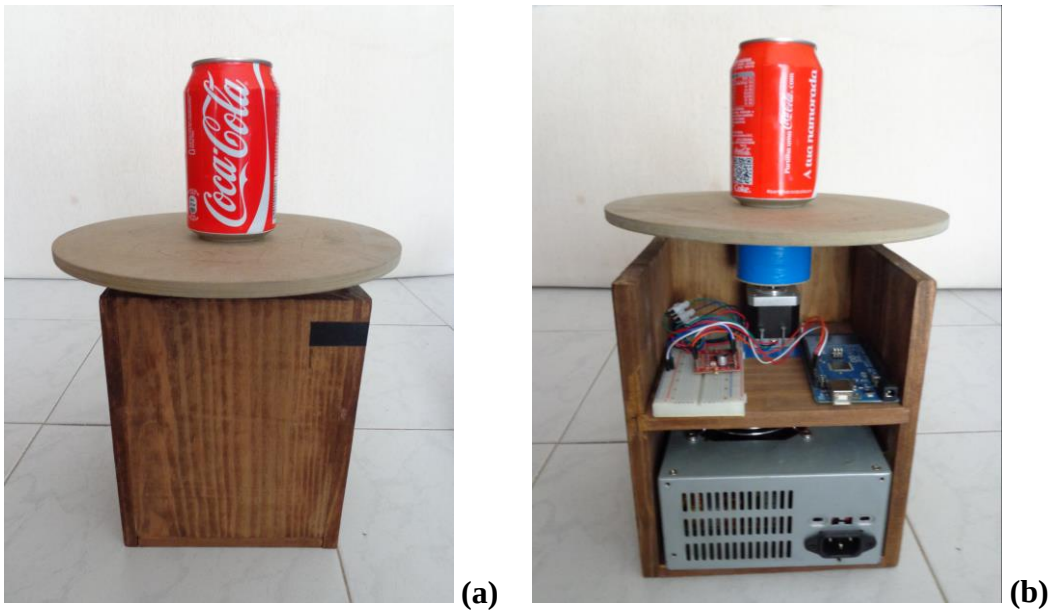


Figura 59: Esquema de ligações do motor de passo, *Big Easy Driver* e *Arduino Mega*

Para se controlar a velocidade do motor de passo é enviado uma sequência de pulsos digitais num determinado intervalo de tempo. Quanto menor for o intervalo, maior será a velocidade a que o motor irá rodar. O pino 13 (*STEP*) é alternado com valores altos e baixo entre os diversos pulsos digitais transmitidos num determinado período. A placa *Big Easy Driver* por defeito é definida no modo de 16 *microsteps* por passo, logo será necessário 3200 passos para uma rotação completa de um motor de 200 passos (*360º/1.8º*). Este processo de *microstepping* permite um controlo mais suave e silencioso, com melhor precisão para velocidades inferiores. Para além disso, a direção do motor pode ser modificada a partir da inversão do valor do pulso digital enviado. Neste caso, como é necessário que a direção não se altere, o pino 12 (*DIR*) será definido com o valor lógico 1 (*5V*) durante toda a operação. Como tal, a plataforma rotativa irá então rodar no sentido dos ponteiros do relógio.

5.5 Implementação

Neste capítulo, é descrito um sistema que captura uma sequência de nuvens de pontos de um objeto que roda em torno de um único eixo de forma automática. O ângulo de rotação entre as diferentes vistas não é ambíguo. O motor de passo utilizado na plataforma rotativa foi definido para rodar *10º* a cada 3 segundos até perfazer a volta completa (*360º*). Resumindo, para obter as diversas capturas de um determinado objeto (36 capturas para cada) precisamos de 1 min 50 s para completar a totalidade do processo de aquisição de nuvens de pontos. A partir da estrutura e dos diversos componentes mencionados anteriormente, a construção da plataforma rotativa resultante pode ser exibida na *Figura 60*.



(a) **(b)**

Figura 60: Plataforma rotativa: vista frontal (a) vista da retaguarda (b)

6. APLICAÇÃO

Neste capítulo é descrito as principais características da aplicação desenvolvida neste projeto de investigação para a digitalização de objetos de pequenas dimensões. Por fim, são ilustradas algumas imagens do funcionamento deste sistema, bem como da abertura dos ficheiros obtidos por esta aplicação num programa de CAD.

6.1 Caraterísticas da aplicação

A arquitetura do sistema de digitalização *3D* de pequenos objetos que implementa o registo de nuvens de pontos e, posteriormente o seu processamento e reconstrução numa malha tridimensional é descrita nesta secção. Assim sendo, o objetivo desta aplicação é o registo de nuvens de pontos adquiridas pelo sensor *Kinect* por forma a resultar um modelo global capaz de representar o objeto na sua totalidade. A partir do modelo resultante, este será processado e reconstruído com a finalidade de gerar uma malha tridimensional que poderá ser lida e editada em programas de *CAD*.

Desta forma, a aplicação desenvolvida no âmbito desta dissertação permite:

- Escolha do nome da pasta e dos ficheiros que serão guardados (em formato *.pcd*);
- Visualização do objeto colocado sobre a plataforma rotativa e da cena que o rodeia;
- Filtragem e segmentação do ambiente envolvente ao objeto (em tempo real);
- Gravação de diversas capturas de um objeto (ordenadas por ordem de entrada);
- Leitura das nuvens de pontos adquiridas pelo sistema implementado;
- Processo de registo de nuvens de pontos previamente adquiridas pelo sensor;
- Visualização do modelo resultante em cada iteração da operação de registo;
- Processamento e reconstrução de um modelo resultante (escolhido pelo utilizador);
- Gravação da malha tridimensional obtida em formato *.ply*, *.obj*, *.stl*.

As *Figuras 61 e 62* ilustram o funcionamento do sistema durante o processo de registo de nuvens de pontos e de processamento/reconstrução do modelo resultante, respetivamente.

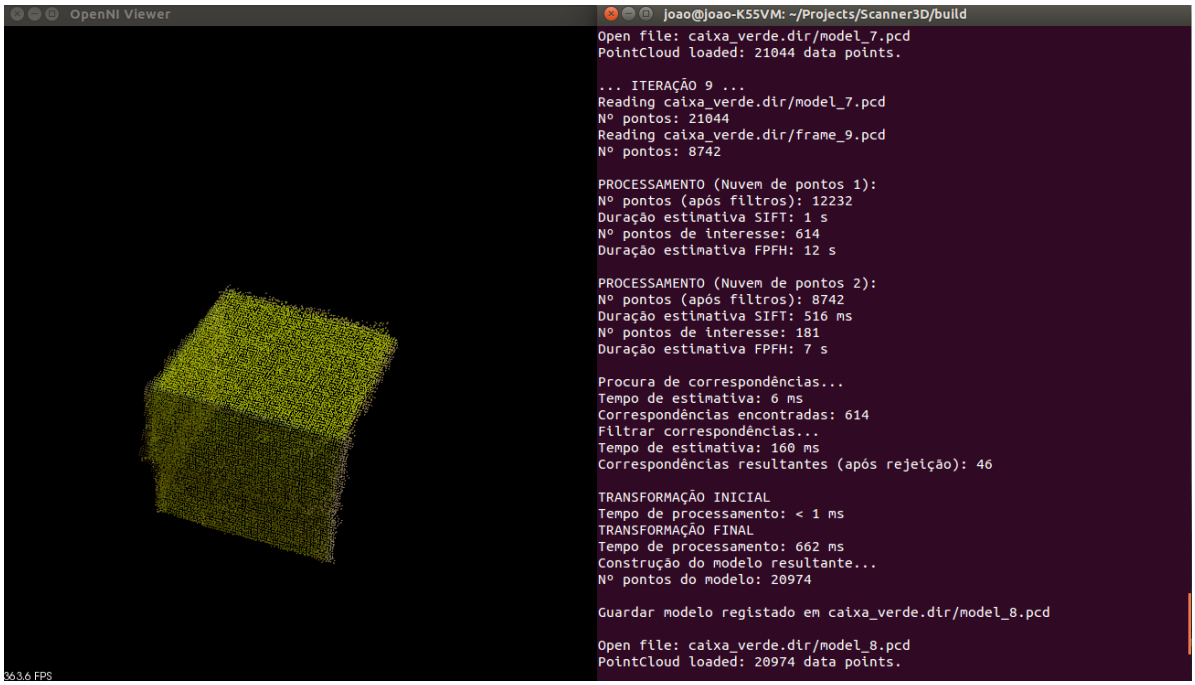


Figura 61: Processo de registo de nuvens de pontos

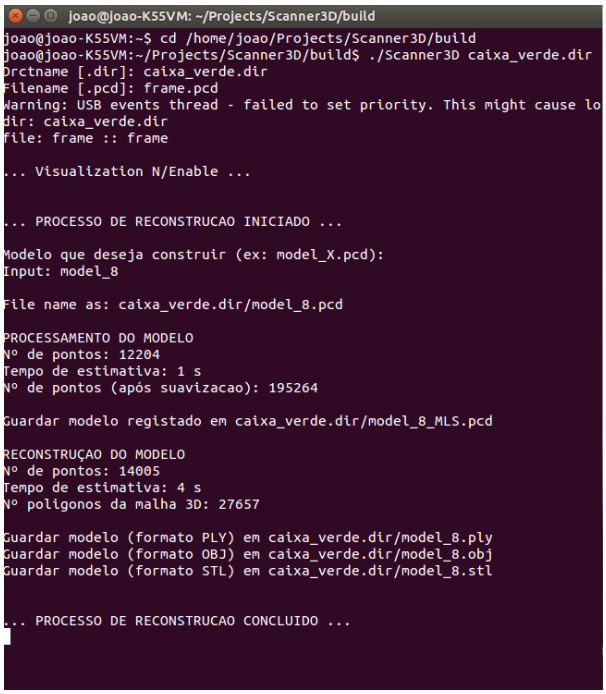
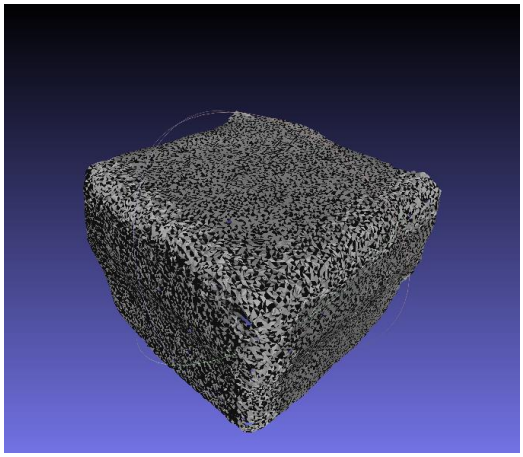


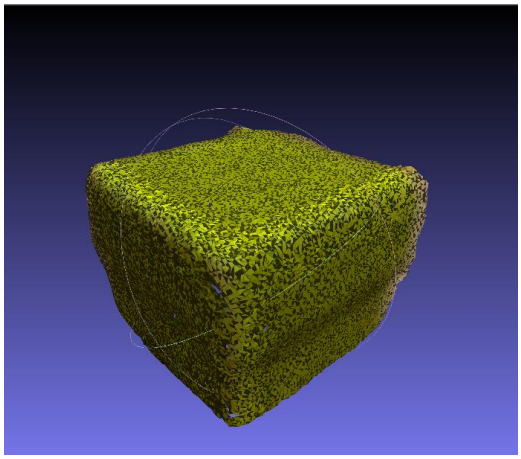
Figura 62: Processamento e reconstrução de um modelo

Pela visualização das duas figuras anteriores é possível identificar os dados mais importantes bem como o tempo de processamento dos diferentes algoritmos implementados em ambos os processos.

A *Figura 63* apresenta a leitura de uma malha tridimensional resultante obtida através desta aplicação num programa de *CAD*, neste caso foi utilizado o programa *MeshLab*.



(a)



(b)

Figura 63: Leitura da malha tridimensional obtida no programa *MeshLab* em formato *.stl* (a) e em formato *.ply* (b)

7. RESULTADOS E DISCUSSÃO

No presente capítulo é apresentado um conjunto de experiências e estudos por forma a avaliar a influência dos parâmetros dos algoritmos responsáveis pelo processo de registo de nuvens de pontos e de reconstrução dos modelos originados. A comparação e discussão destes resultados é descrita em cada etapa, e por fim é feita uma avaliação global dos resultados alcançados por esta aplicação.

7.1 Introdução

Neste capítulo será descrito a organização das diferentes etapas e o relacionamento entre elas por forma a possibilitar a digitalização completa de um objeto e a criação de um modelo final. Todos os módulos e algoritmos associados a este sistema tiveram como base a fundamentação teórica do *capítulo 3*. A partir de um conjunto de nuvens de pontos adquiridas pelo sensor *Microsoft Kinect* foi testada a reconstrução tridimensional de um modelo global capaz de caracterizar um determinado objeto. Um conjunto de testes em objetos com diferentes tamanhos, características geométricas e texturas foram realizados com o objetivo de analisar e verificar a credibilidade de cada algoritmo desta aplicação. Nestas experiências, os melhores resultados obtidos para os diversos conjuntos de parâmetros foram previamente definidos no código da aplicação. No entanto, a alteração dos parâmetros dos diversos algoritmos utilizados neste código pode influenciar negativamente o modelo resultante deste programa, e como tal comprometer o processo de registo de nuvens de pontos.

A *Figura 64* ilustra alguns dos objetos que foram digitalizados a partir do sensor *Kinect* e, por sua vez presentes nas nuvens de pontos utilizadas para avaliar o processo de registo.

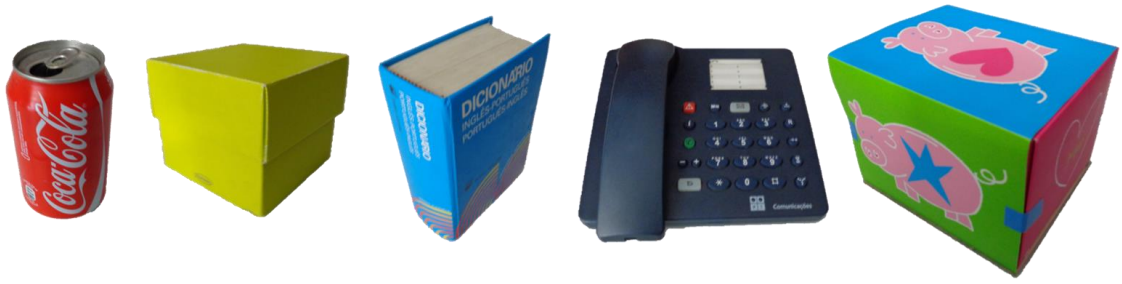


Figura 64: Objetos utilizados no registo de nuvens de pontos

7.2 Aquisição dos dados

Como o sensor *Microsoft Kinect* utiliza a técnica de luz estruturada para medições de profundidade, isto é luz infravermelha, este dispositivo é considerado ideal para ambientes *indoor*. Desta forma, é fundamental encontrar a correta posição para posicionar o sensor de forma a ser possível digitalizar um objeto. Para tal, um objeto colocado na plataforma rotativa, com dimensões 16x13x12 cm (*larg/comp/alt*), foi medido a diversas distâncias no campo de operação do sensor *Kinect* desde 0,45 m a 1,0 m com intervalos de 10 cm. A *Tabela 11* descreve os resultados obtidos durante este estudo.

Tabela 11: Caracterização dos resultados dos testes de profundidade

Profundidade	Resultado do ensaio
45 cm	*Mau. Superfície frontal do objeto não detetada.*
55 cm	*Razoável. Presença de pequenos buracos na superfície frontal do objeto.*
65 cm	*Bom. Objeto corretamente visualizado, nuvem de pontos em boas condições.*
+65 cm	*Bom. Resultado igual ao descrito na condição anterior, no entanto não se justifica um maior afastamento do objeto.*

Com esta experiência conclui-se que a partir dos 55 cm todas as nuvens de pontos são capturadas em boas condições, embora para esta distância com uma certa elevação sejam visualizados uns pequenos erros na superfície do objeto em análise. Assim sendo, a partir do melhor resultado de profundidade encontrado foi testado qual a melhor altura capaz de se conseguir digitalizar numa única captura a superfície frontal e superior deste objeto. A *Tabela 12* descreve os resultados obtidos durante este estudo.

Tabela 12: Caracterização dos resultados dos testes de altura

Altura	Resultado do ensaio
55 cm	Mau. Parte superior do objeto não detetada.
65 cm	Razoável. Presença de buracos na parte superior do objeto.
75 cm	Bom. Objeto totalmente visualizado, nuvem de pontos em boas condições.
85 cm	Mau. Objeto não é totalmente visualizado, para isso o ângulo necessita de ser inferior ao limite mínimo do sensor *Kinect*.

Através da análise dos resultados, a melhor altura encontrada é de 75 cm. Nesta situação, os pontos situados no topo da superfície do objeto foram adquiridos com sucesso. Para valores inferiores, não foi possível observar corretamente a superfície do objeto em questão (*Figura 65*). Em sentido contrário, para valores superiores com a mesma profundidade encontrada pelos testes anteriores, não foi possível observar a totalidade do objeto. Para tal, o ângulo do sensor *Kinect* necessita de ser inferior a -27º, que foi o valor utilizado na condição anterior. Como tal, a solução para este problema seria afastar ligeiramente a plataforma e o objeto para se proceder à digitalização.

Figura 65: Nuvem de pontos capturada pelo sensor *Kinect* a 65cm de altura

Para o melhor valor de altura e profundidade encontrado, foi definido uma janela de visualização através da utilização do filtro *Passthrought* em cada um dos eixos, com 0,5 m no eixo de coordenadas *x* e *y* (caso o sensor esteja centrado com a plataforma [-0.25 m, +0.25 m]), e com 1,0 m no eixo de coordenadas *z*. O objetivo da aplicação deste filtro é rejeitar todos os pontos que se encontram fora destes limites, definindo um região de interesse, no qual se inclui o objeto e o plano da plataforma. Contudo, caso seja necessário para objetos de dimensões superiores estes valores podem ser alterados, ou então pode-se afastar a plataforma do sensor e ajustar o ângulo. No entanto, para objetos de pequenas dimensões não se recomenda afastar a plataforma ou alterar as condições descritas, pois os resultados serão afetados.

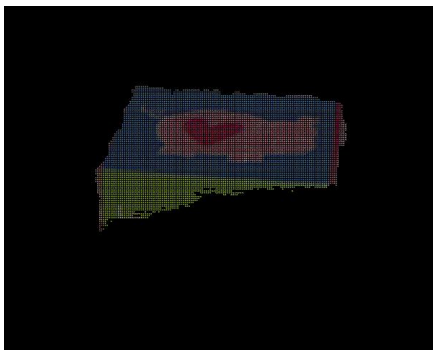
7.3 Segmentação dos objetos

Como referido anteriormente, para o algoritmo de segmentação de planos foi escolhido o método *RANSAC* e o modelo de planos paralelos (*pcl::SACMODEL_PARALLEL_PLANE*). De forma a avaliar este algoritmo, um estudo foi realizado para concluir a influência do valor de distância na segmentação do plano que contém o objeto. Neste estudo, o número máximo de iterações foi definido como 10000. Na *Tabela 13* é descrito os resultados obtidos na segmentação do plano para diferentes valores de distância.

Tabela 13: Caracterização da segmentação do plano para diferentes valores de distância

Distância (m)	Resultado do teste
0.05	Mau. Plano que contém o objeto não foi corretamente segmentado, parte frontal do objeto parcialmente segmentada.
0.01	Bom. Objeto corretamente segmentado, apesar de existirem pequenos pontos do plano nos contornos do objeto.
0.005	Bom. Objeto corretamente segmentado, apesar de existirem pequenos pontos do plano nos contornos do objeto.
0.002	Mau. Plano que contém o objeto não foi segmentado corretamente, existência de ligeiros pontos do plano na nuvem de pontos capturada.
0.001	Mau. Plano que contém o objeto não foi segmentado corretamente, existência de muitos pontos do plano na nuvem de pontos capturada.

A *Figura 66* ilustra os piores casos obtidos para um valor de distância muito elevado e baixo, respetivamente. Pela análise da tabela anterior, é possível concluir que os melhores resultados foram obtidos para distâncias de 0,01 m e 0,005 m. No entanto, em ambos os casos é percetível a existência de pequenos pontos do plano nas extremidades dos objetos. Para reduzir esta quantidade de pontos é utilizado o algoritmo de remoção de *outliers*. Contudo, como o tempo de processamento deste algoritmo é bastante reduzido, a segmentação do plano e extração do *cluster* que contém o objeto pode ser realizada em tempo real.



(a)

(b)

Figura 66: Segmentação de planos para valores de distância de 0.05 m (a) e de 0.001 m (b)

Um dos parâmetros mais importantes na extração do objeto é a tolerância do *cluster*. Este parâmetro deve ser superior à resolução da nuvem de pontos capturada pelo sensor *Kinect*. Como a resolução da nuvem de pontos é sensivelmente 0,001 m, o estudo deste parâmetro foi avaliado entre 0,001 e 0,1. Assim sendo, para o valor mínimo não foi possível extrair nenhum *cluster* desse conjunto de dados. Após diversas experiências com diferentes objetos, a melhor configuração para este parâmetro é 0,005 m. Por outro lado, se este valor for superior, maior será o tempo de processamento e a probabilidade de resultados incorretos. Por fim, resta salientar o facto de neste estudo o tamanho mínimo e máximo do *cluster* ter sido definido por 5.000 e 30.000 pontos, respetivamente. A *Figura 67* ilustra a segmentação do plano e extração do *cluster* de dois objetos de diferentes tamanhos.


(a)

(b)

Figura 67: Segmentação de objeto de pequena dimensão (a) e de grande dimensão (b)

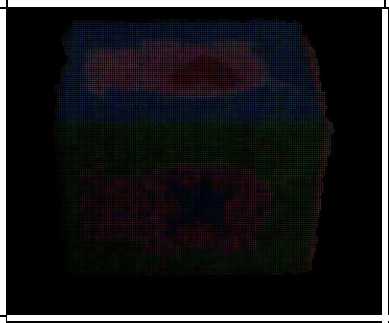
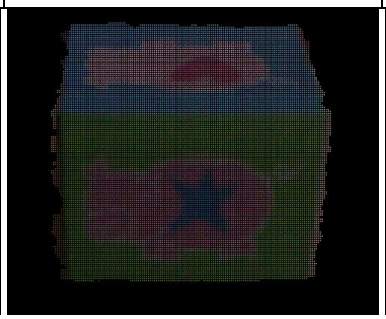
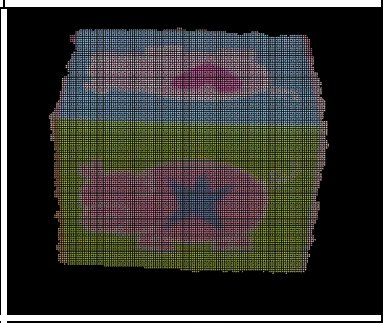
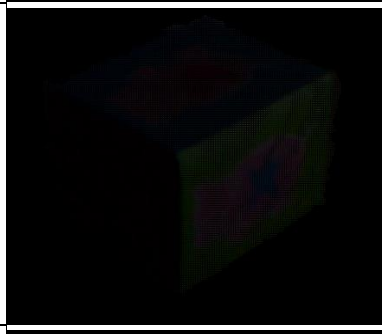
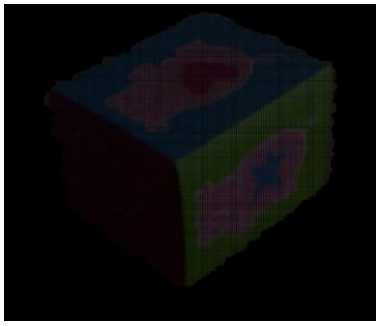
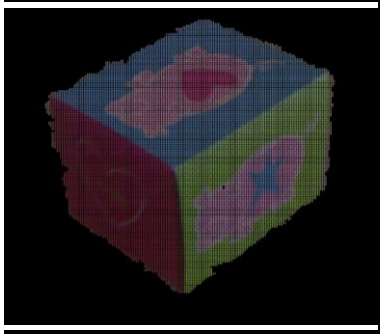
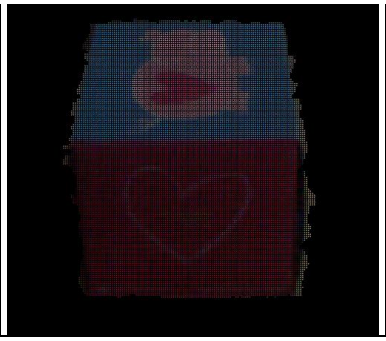
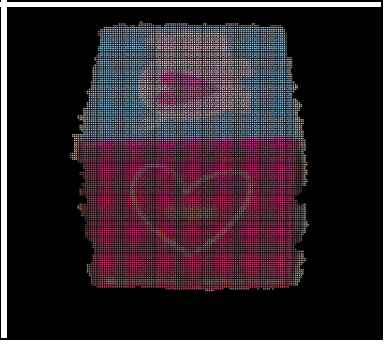
Na *Figura 67-a*, pode ser vista a segmentação do plano de um objeto de pequenas dimensões. Embora permaneça a presença de diversos pontos nas extremidades do objeto, é possível observar que existe um maior erro de segmentação nessas mesmas extremidades, onde pequenas regiões do objeto são excluídas. Assim sendo, o sistema tem uma dificuldade acrescida na correta segmentação de objetos de dimensões inferiores.

Na *Figura 67-b*, embora seja um objeto de dimensões superiores comparado com o anterior, o pequeno raio da cabeça do objeto torna mais complicado para o sistema distinguir os pontos válidos desta área da nuvem de pontos capturada pelo sensor. Portanto, características geométricas específicas de tamanho reduzido ligeiramente isolado do objeto podem não ser interpretadas por este algoritmo em todas as iterações e como tal serem excluídas do *cluster*, por exemplo uma caneca.

Porém, o tamanho do plano também influencia o resultado da segmentação. A base que contém o objeto deve ser duas vezes superior ao comprimento e à largura do objeto em análise, caso contrário não é possível segmentar corretamente o plano do objeto. Neste conjunto de testes, foi aplicado à plataforma rotativa uma base de maior dimensão para os testes com objetos com dimensões superior a 15 cm.

Por último, foi verificada a influência das condições de luminosidade na segmentação de objetos. Como referido anteriormente, o sensor *Microsoft Kinect* é altamente sensível às condições de luminosidade e à natureza da superfície dos objetos. Na *Tabela 14* é apresentado os resultados obtidos na segmentação do um objeto para três condições de luminosidade em três posições diferentes.

Tabela 14: Segmentação de objetos com diferentes condições de luminosidade

	Luminosidade Baixa	**Luminosidade Média**	**Luminosidade Alta**
Posição 1			
Posição 2			
Posição 3			

Para condições de luminosidade baixas, a qualidade dos dados da nuvem de pontos capturada são inferiores. Nas posições 1 e 3 desta condição, é possível reparar na elevada quantidade de pontos sem informação na superfície do objeto. Contudo, este problema deve-se a uma má aquisição dos dados de profundidade pelo sensor *Kinect* para condições de baixa luminosidade. No entanto, os resultados da segmentação são idênticos aos obtidos nas outras condições de luminosidade, embora haja um maior número de irregularidades na superfície frontal do objeto. As diferenças entre as condições de luminosidade média e alta são irrelevantes e inconclusivas.

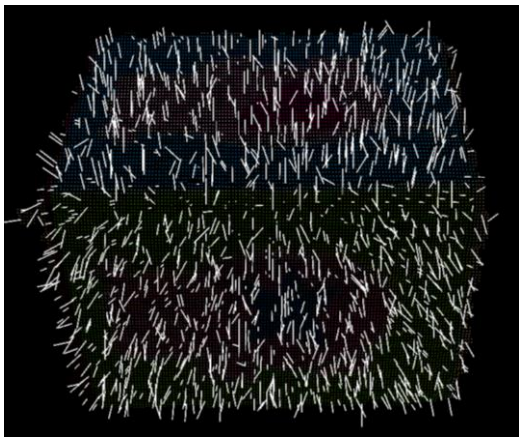
7.4 Estimativa das normais

Através da utilização do algoritmo implementado na classe *pcl::NormalEstimationOMP* é possível calcular as normais da superfície de um objeto de forma bastante rápida em relação ao algoritmo de estimativa original. Na *Tabela 15*, é descrito um estudo realizado por forma a verificar de que modo a escala do raio de procura da vizinhança para um determinado ponto afeta o resultado da estimativa das normais. O tempo de processamento deste algoritmo para uma nuvem de pontos inicialmente filtrada e segmentada com cerca de 15.000 pontos é também referido nesta tabela.

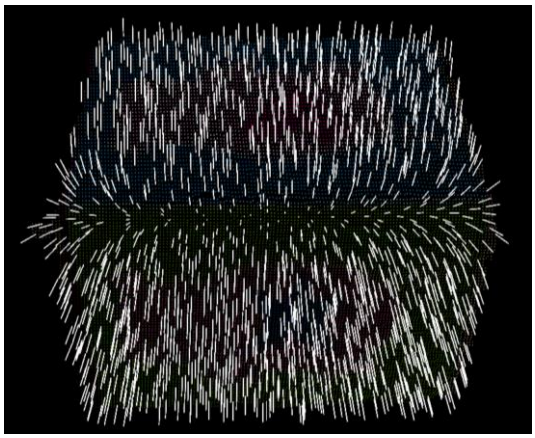
Tabela 15: Caracterização do raio de procura na estimativa das normais

Raio de estimativa	Tempo de processamento	Resultado
0.001	*28 ms*	Impossível. O algoritmo não consegue estimar as normais num raio tão pequeno.
0.005	*62 ms*	Mau. Estimativa incorreta em quase toda a nuvem de pontos.
0.01	*149 ms*	Mau. Estimativa ligeiramente distorcida.
0.03	*1,1 s*	Bom. Estimativa praticamente correta em toda a nuvem de pontos.
0.05	*3,1 s*	Bom. Resultado idêntico à condição anterior, no entanto ligeiramente distorcida em certos locais.
0.1	*9,9 s*	Razoável. Pequenos erros de estimativa nas extremidades do objeto.
0.5	*18,3 s*	Mau. Estimativa ligeiramente distorcida.

Estes resultados permitem concluir que quanto maior for o raio de análise dos vizinhos de determinado ponto maior é o tempo de processamento deste algoritmo. Para valores iguais ou inferiores a 0,001 m é impossível calcular estas características, porque este valor tem de ser superior à resolução da nuvem de pontos. Assim sendo, o raio de estimativa está diretamente relacionado com a densidade da nuvem de pontos. Para valores muito inferiores ou superiores a este nível de detalhe, a estimativa das normais pode resultar em características distorcidas das que realmente se pretendem. A *Figura 68* ilustra o efeito da modificação do raio de estimativa das normais da superfície de um objeto.



(a)



(b)

Figura 68: Estimativa das normais para um raio de estimativa de 0,005 m (a) e 0,03 m (b)

Na *Figura 68-a*, é possível visualizar que a utilização de um raio pequeno para análise da vizinhança resulta em normais distorcidas na maioria dos pontos da superfície deste objeto. Pelo contrário, na *Figura 68-b* após um ajuste deste parâmetro em relação ao nível de detalhe da nuvem de pontos, é possível concluir que a qualidade destas características é muito superior. Como tal, é possível identificar a perpendicularidade destas normais quer no topo quer na parte frontal em relação à superfície do objeto. Assim sendo, as etapas de processamento que dependem destas características podem descrever de forma mais eficiente o objeto presente nas nuvens de pontos.

7.5 Deteção de pontos de interesse

Nesta secção será caracterizado o método de deteção de pontos de interesse pelo algoritmo *Scale Invariant Feature Transform (SIFT)*. Nas *Tabelas 16 e 17*, é feita uma análise da forma como o parâmetro escala mínima afeta este algoritmo a partir de uma nuvem de pontos obtida a partir do sensor *Kinect.* A nuvem de pontos utilizada neste estudo foi inicialmente filtrada e segmentada contendo 15.000 pontos. Pode ser observado que o número de pontos de interesse presentes na nuvem de pontos aumenta com a diminuição da escala mínima. O valor do contraste mínimo afeta a quantidade de pontos de interesse obtidos, no entanto o tempo de processamento é praticamente idêntico apesar da quantidade de pontos ter duplicado. Por outro lado, com a diminuição do contraste mínimo existem alguns pontos de interesse que se encontram dispersos no meio da superfície do objeto. Portanto, um valor de contraste superior garante que a maior parte dos pontos de interesse se encontrem sobre características geométricas salientes de um determinado objeto, tais como contornos e extremidades.

Tabela 16: Resultados do algoritmo *SIFT* para valor de contraste mínimo de 1.0

Escala mínima	Nº pontos de interesse	Tempo de processamento	Resultado
0.001	*668*	*610 ms*	*Muito bom*
0.002	*291*	*385 ms*	*Bom*
0.005	*72*	*100 ms*	*Mau*
0.01	*25*	*29 ms*	*Mau*
0.05	*1*	*1 ms*	*Mau*

Tabela 17: Resultados do algoritmo *SIFT* para valor de contraste mínimo de 0.1

Escala mínima	Nº pontos de interesse	Tempo de processamento	Resultado
0.001	*1503*	*615 ms*	*Bom*
0.002	*481*	*388 ms*	*Razoável*
0.005	*88*	*95 ms*	*Mau*
0.01	*27*	*27 ms*	*Mau*
0.05	*1*	*1 ms*	*Mau*

7.6 Descritores de características

Nesta secção será caracterizado o método de estimativa dos descritores geométricos com base no algoritmo de estimativa *FPFH*. A *Tabela 18* descreve o comportamento do número de correspondências estimadas e resultantes (após terem sido rejeitadas) com a modificação do raio de estimativa do descritor das características, mantendo todos os outros parâmetros constantes. Para valores de raio muito elevados e muito baixos podem ser originadas más correspondências. Ao contrário do parâmetro escala mínima entre os pontos de interesse, onde uma diminuição deste valor representa um aumento da quantidade de correspondências, os descritores de características têm uma relação diferente. Para valores inferiores a 0,05 m, a qualidade das correspondências foi afetada de tal modo que levou à ocorrência de falhas. Assim sendo, quanto maior for o valor do raio de estimativa melhor serão as correspondências encontradas. No entanto, o aumento deste raio influencia o tempo de processamento deste algoritmo. Neste caso, entre duas nuvens de pontos consecutivas (com um número de pontos idêntico) desfasadas *10º* entre si, o problema não é muito perturbador. Contudo, após a concatenação destes conjuntos num processo de registo de nuvens de pontos, o que traduz num aumento da quantidade de pontos, o resultado é diferente. Como tal, tem de se ter em atenção ao detalhe das nuvens de pontos na escolha do melhor raio de estimativa dos descritores de características.

Tabela 18: Efeito do raio dos descritores no processamento de uma nuvem de pontos

Raio da estimativa	**Tempo de processamento**	**Correspondências**	
		Estimadas	**Após Rejeição**
0.01	*204 ms*	*668*	*47*
0.03	*1,8 s*	*668*	*138*
0.05	*4,7 s*	*668*	*217*
0.08	*11,3 s*	*668*	*338*
0.1	*15,4 s*	*668*	*379*
0.5	*26,4 s*	*668*	*464*

7.7 Procura e rejeição de correspondências

Após a escolha do parâmetro para o algoritmo de descrição de características, é necessário escolher a distância de estimativa para o algoritmo de rejeição de correspondências. Com esta experiência pretendia-se constatar a influência da distância de estimativa na rejeição das más correspondências pelo método *RANSAC*, assim como o tempo que a operação demora. Este teste foi efetuado entre duas nuvens de pontos sucessivas com um desfasamento de *10º* entre si. Para a procura das correspondências, os efeitos nos dois tipos de estimativa foram avaliados, no método direto e no método reciproco. Os dados da *Tabela 19* ilustram os resultados obtidos na estimativa e rejeição de correspondências na primeira iteração do processo de registo.

Tabela 19: Influência do raio da estimativa na rejeição de correspondências para um ângulo de desfasamento *10º*

Distância de estimativa	**Estimativa das correspondências (método direto)**				**Estimativa das correspondências (método reciproco)**			
	Estimadas	**Tempo**	**Após rejeição**	**Tempo**	**Estimadas**	**Tempo**	**Após rejeição**	**Tempo**
0.001	668	3 ms	21	663 ms	198	7 ms	12	112 ms
0.002	668	3 ms	59	89 ms	198	5 ms	30	8 ms
0.005	668	3 ms	195	2 ms	198	5 ms	87	< 1 ms
0.01	668	3 ms	338	< 1 ms	198	6 ms	122	< 1 ms
0.02	668	3 ms	473	< 1 ms	198	6 ms	164	< 1 ms
0.05	668	3 ms	583	< 1 ms	198	5 ms	190	< 1 ms

Para os valores inferiores a 0.005, as correspondências encontradas nãos são suficientes para garantir o correto alinhamento entre as duas nuvens de pontos. Pela análise dos resultados da *Tabela 19*, é possível concluir que quanto menor for a distância de estimativa maior será o tempo de processamento do algoritmo de rejeição de correspondências. Tal como nos descritores de características, para valores muito elevados e muito baixos podem surgir erros de correspondências. De uma forma geral, para gama de valores mais aconselhados o tempo de processamento deste algoritmo é muito baixo. O tempo de processamento na estimativa das correspondências é superior para o método reciproco, isto deve-se ao fato de serem testadas as correspondências em ambas as nuvens de pontos, logo o tempo será duas vezes maior que no método direto.

De seguida, foi testado a influência do ângulo de desfasamento entre as duas nuvens de pontos. A *Tabela 20* descreve os resultados obtidos para as mesmas condições definidas no estudo anterior, mas com uma transformação superior entre as duas nuvens de pontos, neste caso o dobro do valor definido anteriormente, *20º*.

Tabela 20: Influência do raio da estimativa na rejeição de correspondências para um ângulo de desfasamento *20º*

Distância de estimativa	Estimativa das correspondências (método direto)				Estimativa das correspondências (método reciproco)			
	Estimadas	Tempo	Após rejeição	Tempo	Estimadas	Tempo	Após rejeição	Tempo
0.001	668	3 ms	8	668 ms	96	6 ms	6	72 ms
0.002	668	4 ms	20	721 ms	96	7 ms	11	11 ms
0.005	668	3 ms	63	76 ms	96	9 ms	31	< 1 ms
0.01	668	4 ms	127	11 ms	96	7 ms	49	< 1 ms
0.02	668	4 ms	222	1 ms	96	5 ms	74	< 1 ms
0.05	668	2 ms	417	< 1 ms	96	6 ms	92	< 1 ms

Os resultados obtidos foram idênticos aos anteriores. Para os valores do meio da tabela as correspondências obtidas após a rejeição são capazes de descrever corretamente a transformação entre estes dois conjuntos de dados. Para os valores mais baixos e mais elevados, a respetiva escassez e excesso de correspondências pode não ser a opção mais indicada para descrever a transformação, podendo levar a origem de erros. De salientar o facto que com uma maior distância entre os dados as correspondências encontradas após a rejeição foram diminuídas para aproximadamente metade do valor obtido anteriormente. Para além disso, é de ressalvar a ideia que no método reciproco as correspondências estimadas ente as nuvens de pontos terem sido reduzidas para metade.

Por fim, verificou-se que para desfasamentos superiores a *20º* os valores encontrados dificilmente serão capazes de conseguir descrever a transformação entre os dois conjuntos.

7.8 Registo de nuvens de pontos

Uma boa estimativa das normais da superfície de um objeto e uma boa descrição das características dos pontos de interesse têm uma grande influência na estimativa das correspondências entre nuvens de pontos. A partir dos melhores resultados alcançados nos diversos testes realizados anteriormente, foi avaliado a influência destes valores no alinhamento inicial e refinado de modelos adquiridos no processo de registo de nuvens de pontos.

No que respeita ao alinhamento inicial, dois métodos foram implementados e avaliados por forma a escolher a melhor opção para este processo. Na biblioteca *PCL*, estes dois algoritmos são denominados *TransformationEstimationSVD* e *TransformationEstimationLM*.

A primeira solução implementada calcula a transformação rígida entre nuvens de pontos a partir da decomposição de um valor único, denominada *SVD*. Esta transformação é estimada a partir das correspondências obtidas entre os pontos de interesse de cada nuvem de pontos. Os resultados obtidos através deste algoritmo, após diversos testes com diferentes objetos, sugerem que a estimativa da transformação rígida entre nuvens de pontos é corretamente atingida. Contudo, como este método é uma solução ponto-a-ponto, a estimativa é realizada tendo em conta a interseção desses pontos correspondentes em ambos os conjunto de dados. Assim sendo, caso as correspondências encontradas não estiverem bem distribuídas pelo objeto podem ocorrer pequenos erros de alinhamento que prejudicam o modelo resultante, tais como ligeiras inclinações e pequenos desfasamentos em certas regiões.

A outra solução implementada é um método iterativo baseado em mínimos quadrados do algoritmo padrão *Levenberg-Marquardt*. Assim como o algoritmo descrito anteriormente, esta transformação é estimada a partir das correspondências obtidas entre os pontos de interesse de cada nuvem de pontos. O estudo realizado com esta técnica mostrou uma melhor adaptação às modificações ocorridas em cada iteração do que no algoritmo anterior. Todavia, a utilização de soluções com base na distância ponto-ao-plano em vez de ponto-a-ponto permite que as regiões planas deslizem ao longo umas das outras, nos casos em que as correspondências não são as mais recomendáveis.

Em termos qualitativos os dois métodos produzem boas estimativas da transformação inicial, mas o método baseado no algoritmo *Levenberg-Marquardt*, necessita em geral de menos tempo de processamento para produzir bons resultados.

De seguida, foi avaliado o algoritmo responsável pelo alinhamento refinado, o *ICP*. O algoritmo *ICP* representa um método iterativo que permite encontrar a transformação entre duas nuvens de pontos através da minimização do erro da distância entre as regiões que se sobrepõem obtidas anteriormente no alinhamento inicial. Para este algoritmo, o número máximo de iterações e a variação *épsilon* (ε) foram estabelecidos com os valores 10.000 e $1{,}0 * 10^{-6}$, respetivamente. Estes dois parâmetros são responsáveis por definir os critérios de convergência e de término do processo. Além disso, é preciso estabelecer a distância entre correspondências e a distância do ciclo interno de rejeição de *outliers*.

A avaliação da qualidade dos resultados deste algoritmo de refinamento é difícil de se conseguir, no entanto a validação dos seus efeitos foi realizada visualmente. Como descrito em algoritmos utilizados anteriormente, o valor destes parâmetros está diretamente relacionado com a resolução e densidade das nuvens de pontos. Assim sendo, para ambas as distâncias valores inferiores a 0,005 e superiores a 0,05 tiveram como consequência más transformações, logo nas primeiras iterações deste processo. Deste modo, a influência destes parâmetros foi testada para valores no intervalo $[0.005, 0.05]$. Na *Tabela 21*, é enumerado as consequências da modificação destes valores no processo de registo de diversas nuvens de pontos.

Tabela 21*:* Caracterização do alinhamento refinado pelo algoritmo *ICP*

Distância máxima de correspondência	**Distância *RANSAC Outlier Removal***	**Resultado do teste**
0,005	***0,005***	Bom. Alinhamento das nuvens de pontos correto para diversas iterações.
0,005	***0,01***	Bom. Resultado idêntico à condição anterior.
0,005	***0,05***	Mau. Distância para o método RANSAC excessiva. Falha no registo de nuvens de pontos após poucas iterações.
0,01	***0,005***	Bom. Alinhamento das nuvens de pontos correto para diversas iterações.
0,01	***0,01***	Bom. Resultado idêntico à condição anterior.
0,05	***0,01***	Mau. Distância entre correspondências excessiva. Falha no registo de nuvens de pontos após poucas iterações.

Os resultados obtidos na caracterização do algoritmo *ICP* são muito subjetivos. A única conclusão que se pode retirar desta experiência é que para valores *0,01* e *0,005* obtém-se bons alinhamentos entre as nuvens de pontos, apesar de em alguns casos a estimativa inicial não ter sido a mais aconselhável. Portanto, este algoritmo depende diretamente das correspondências

obtidas entre os dois conjuntos de dados e do alinhamento inicial alcançado anteriormente. No caso de as correspondências não terem sido estimadas corretamente, por exemplo se só tiverem sido estimadas correspondências entre pontos no topo da superfície de um objeto, este algoritmo pode convergir para um valor que embora seja aceitável no seu ponto de vista, não seja considerado um bom alinhamento para as nuvens de pontos.

A *Tabela 22* descreve os resultados obtidos nas dez primeiras iterações do processo de registo de nuvens de pontos de um determinado objeto.

Tabela 22: Caracterização do registo de nuvens de pontos para 10 iterações

	Nuvem pontos 1	**PI**	**FPFH**	**Nuvem pontos 2**	**PI**	**FPFH**	**CE**	**CR**	**Alinham. Refinado**	**Pontos do modelo resultante**
1	**14939**	668	13,4 s	**14726**	610	12,7 s	**668**	338	151 ms	**29665**
2	**13891**	626	11,1 s	**14477**	548	12,3 s	**626**	107	723 ms	**28368**
3	**15180**	722	12,6 s	**15633**	662	13,6 s	**722**	157	292 ms	**30813**
4	**17487**	920	16,4 s	**16121**	693	14,2 s	**920**	184	194 ms	**33608**
5	**20092**	1005	20,6 s	**16013**	659	13,4 s	**1005**	297	314 ms	**36105**
6	**21357**	1040	23,4 s	**15544**	678	13,5 s	**1040**	288	266 ms	**36901**
7	**21621**	1110	24,4 s	**13240**	637	11,2 s	**1110**	123	1312 ms	**34861**
8	**20969**	1196	23,8 s	**13434**	591	11,4 s	**1196**	138	1195 ms	**34403**
9	**20928**	1158	21,6 s	**13444**	515	11,0 s	**1158**	118	639 ms	**34372**
10	**20739**	1196	21,6 s	**13467**	502	11,2 s	**1196**	121	756 ms	**34206**

PI – Nº de pontos de interesse encontrados **FPFH –** Tempo de estimativa dos descritores de características	**CE –** Nº de correspondências estimadas **CR –** Nº de correspondências após rejeição

Pela análise da tabela, é possível visualizar que o tempo de processamento na estimativa dos descritores de características depende da quantidade de pontos desse conjunto. Assim, quanto maior for o número de pontos maior será o tempo de operação deste algoritmo. Para além disso, o algoritmo de alinhamento refinado, o *ICP*, tem um grande peso computacional. No entanto, devido à aplicação de métodos de alinhamento inicial descritos anteriormente (neste caso especifico, o método baseado no algoritmo *Levenberg-Marquardt)*, pode-se provar que este procedimento é bastante mais rápido para o registo de nuvens de pontos.

Esta técnica de registo consiste na concatenação constante das nuvens de pontos registadas, em cada iteração da operação, a partir do alinhamento do modelo global com a nuvem de pontos seguinte. Após a primeira iteração, a nuvem de pontos 1 corresponde ao

modelo resultante da etapa anterior. Ao modelo resultante aplicou-se um filtro para diminuir a resolução da nuvem de pontos. Verificou-se que a diminuição do número de pontos das nuvens através do algoritmo de *downsampling* não influencia negativamente a qualidade dos resultados finais. No entanto, a utilização deste filtro tem uma grande influência na velocidade de processamento no processo de registo de nuvens de pontos. Contudo, tem de se ter em atenção este valor, pois a diminuição do nível de detalhe da nuvem de pontos pode dificultar a definição de todos os parâmetros que levam ao processo de registo. A solução passa por atingir um equilíbrio entre a perda de detalhe e a velocidade de processamento, tendo em conta que os parâmetros podem divergir em cada iteração. Neste caso, para as dez primeiras iterações do processo de registo de nuvens de pontos foram encontradas bons resultados no alinhamento das diversas nuvens capturada, sendo o modelo capaz de caracterizar o objeto sem qualquer erro associado. A *Figura 69* ilustra o modelo resultante após terem ocorrido dez iterações no processo de registo de nuvens de pontos.

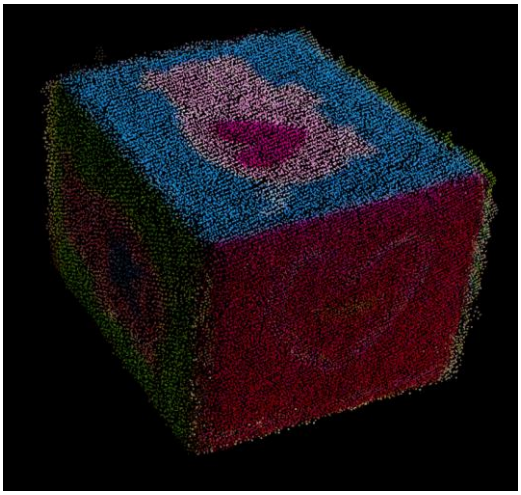


Figura 69: Modelo resultante obtido após dez iterações no processo de registo

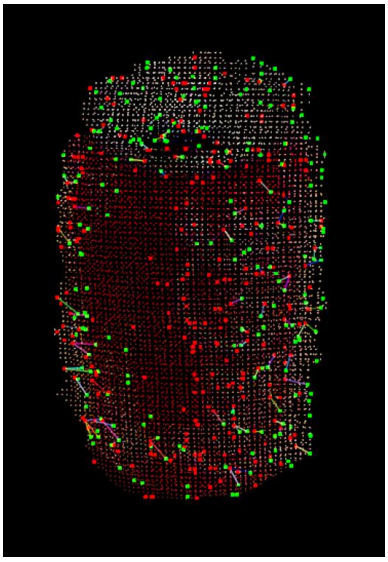
7.9 Discussão dos resultados do processo de registo

Neste capítulo, são apresentados os resultados obtidos a partir do sistema implementado nesta dissertação para a digitalização tridimensional de objetos. Os diversos cenários têm como base os objetos ilustrados no início deste capítulo. A escolha destes objetos teve como preocupação a diversidade de textura, dimensão e características geométricas dos mesmos. A configuração desta aplicação é sustentada pelos resultados alcançados nas diversas experiências efetuadas. Na *Tabela 23* são enumerados os parâmetros de todos os algoritmos utilizados nesta aplicação.

Tabela 23: Caraterização dos parâmetros da aplicação

Algoritmo	Parâmetro	Valor
Filtro Passthrough	*Eixo x (Xmin, Xmax)*	*-0.25, +0.25*
	Eixo y (Ymin, Ymax)	*-0.40, +0.10*
	Eixo z (Zmin, Zmáx)	*0.0, 1.0*
Segmentação do plano e Extração de clusters	*Número máximo de iterações*	*10000*
	Distância limite	*0.01*
	Tamanho mínimo do cluster	*2500*
	Tamanho máximo do cluster	*50000*
	Tolerância do cluster	*0.01*
Filtro Downsampling	*Tamanho do voxel*	*0.002*
Radius Outlier Removal	*Raio de estimativa*	*0.02*
	Número mínimo de vizinhos	*15*
Normal Estimation	*Raio de estimativa*	*0.03*
SIFT Keypoints	*Escala mínima*	*0.001*
	Número de oitavas	*4*
	Número de escala por oitava	*5*
	Contraste mínimo	*1.0*
FPFH Estimation	*Raio de estimativa*	*0.08*
Rejeição de más correspondências	*Número máximo de iterações*	*10000*
	Distância de estimativa	*0.005*
Alinhamento Refinado ICP	*Distância máxima de correspondência*	*0.005*
	Distância RANSAC Outlier Removal	*0.005*
	Variação épsilon	$1{,}0 * 10^{-6}$
	Número máximo de iterações	*10000*

Em todos os testes, os parâmetros dos diversos algoritmos implementados são constantes e têm os valores especificados na tabela anterior. Nas *Figuras 70, 71, 72 e 73* estão ilustrados os resultados práticos do processo de registo de nuvens de pontos implementado nesta aplicação para alguns dos objetos testados. Nestas figuras, podem ser observados os pontos de interesse e correspondências para as duas nuvens de pontos, sendo representado com a cor verde os pontos de interesse da nuvem de origem ou do modelo resultante e a cor vermelha os pontos de interesse da nuvem seguinte. Para uma melhor visualização dos resultados da estimativa das correspondências as duas nuvens de pontos foram sobrepostas na mesma imagem. A primeira situação que foi avaliada tem como base um objeto de pequenas dimensões, neste caso uma lata de um refrigerante.


(a)

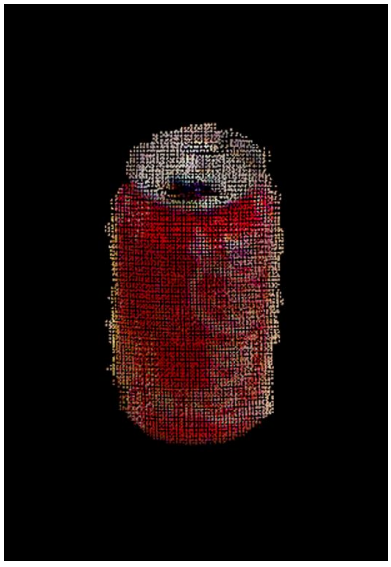

(b)

Figura 70: Teste do objeto 1 – Resultado da estimativa das correspondências (a) e do alinhamento das nuvens de pontos após 5 iterações (b)

Para esta situação verificou-se que os pontos de interesse detetados em ambas as nuvens de pontos não representam com precisão as características geométricas do objeto em questão. Com tal, é possível visualizar diversos pontos dispersos em ambos os conjuntos de dados (*Figura 70-a*). Como consequência, o resultado do processo de registo para várias iterações é a sobreposição de ambas as nuvens de pontos independentemente das correspondências encontradas (*Figura 70-b*). Este erro de alinhamento deve-se à má caracterização dos parâmetros de deteção de pontos de interesse para objetos de dimensões reduzidas, como o representado na figura anterior. Com este teste concluiu-se que para objetos de dimensões

inferiores a 10 cm de largura e de altura esta aplicação não consegue proceder à reconstrução tridimensional desse objeto.

Nas próximas situações, são avaliados os efeitos do alinhamento de nuvens de pontos para objetos de dimensões superiores ao testado anteriormente. Neste cenário específico, o objetivo era determinar o resultado do processo de registo de nuvens de pontos para um objeto com características semelhantes e com uma textura uniforme por toda a sua superfície.

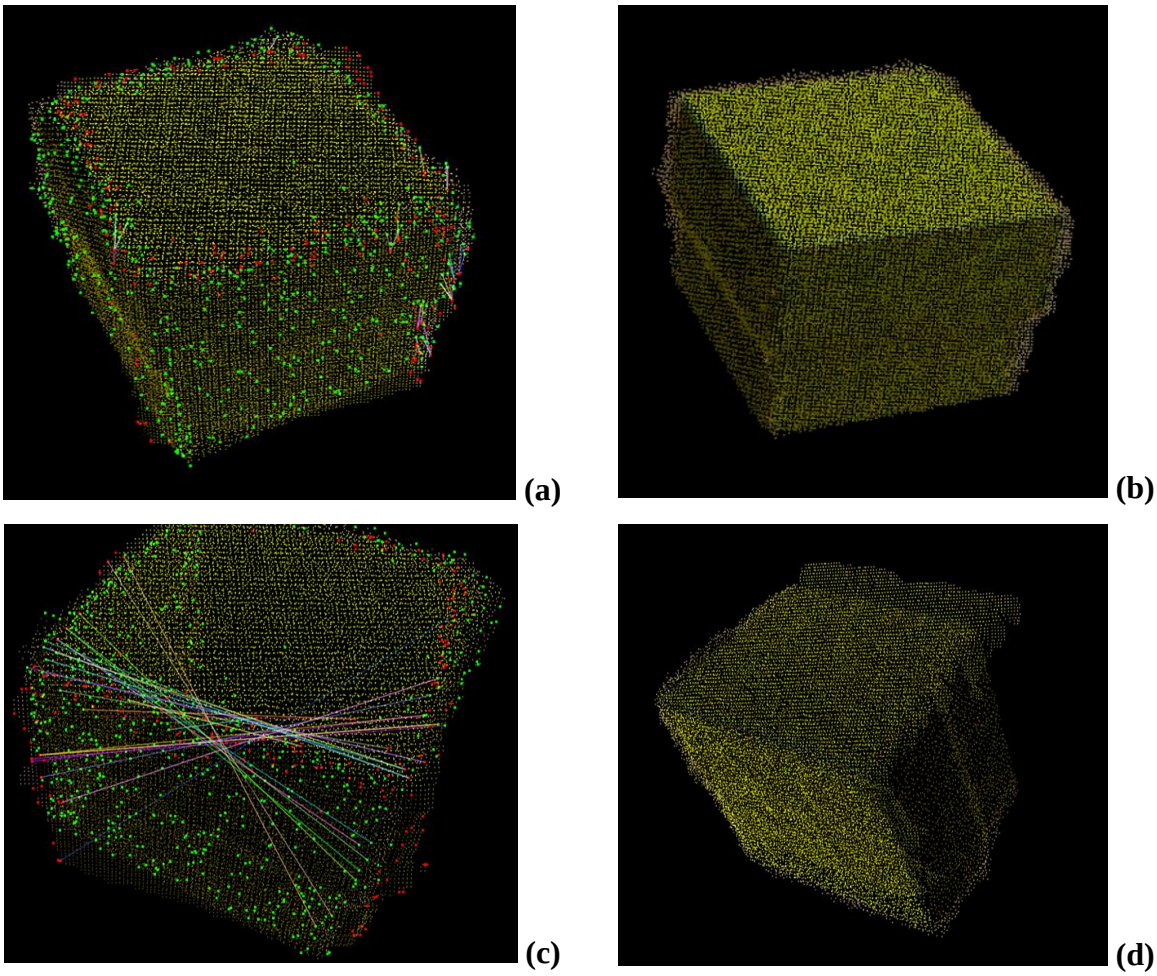
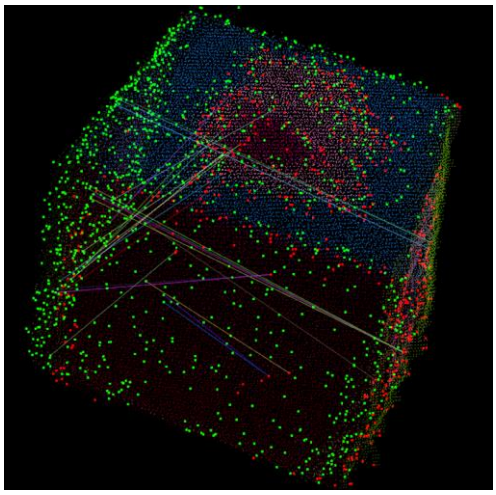


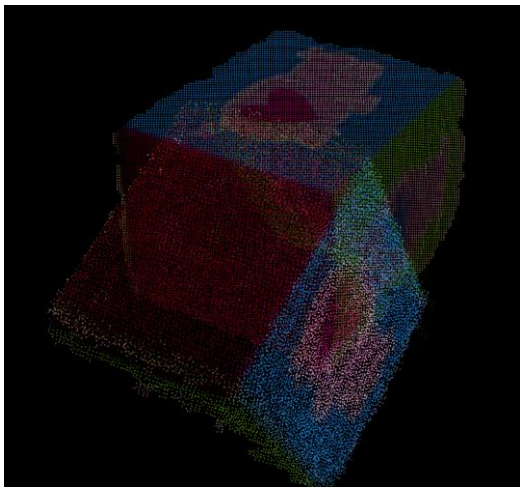
Figura 71: Teste do objeto 2 – Bom resultado da estimativa das correspondências (a) e do alinhamento das nuvens de pontos (b); Mau resultado da estimativa das correspondência (c) e do alinhamento de nuvens de pontos após o dobro das iterações (d)

Na *Figura 71-a* pode ser visualizada uma boa deteção dos pontos de interesse em torno dos contornos do objeto para a nuvem de pontos de entrada (representados com a cor vermelha) e, o mesmo se pode concluir para a nuvem de pontos do modelo resultante da iteração anterior (representados com a cor verde), apesar de em algumas regiões se encontrem pontos dispersos. Em relação às correspondências encontradas, após rejeição resultaram apenas aquelas cujas relações estão situadas nos contornos do objeto. Como tal, o modelo obtido foi corretamente

descrito por estas correspondências (*Figura 71-b*). Contudo, após 15 iterações no processo de registo de nuvens de pontos o resultado não foi o esperado. As correspondências encontradas nesta iteração não correspondem às verdadeiras relações entre ambas as nuvens de pontos. Assim sendo, o modelo resultante é comprometido por este erro e verifica-se uma falha das seguintes etapas do processo de registo. Para verificar se este erro teve origem devido às características geométricas do objeto, ou seja, a textura da superfície e a dimensão do mesmo, foi realizado um teste num outro objeto de maior dimensão e com características mais diversificadas.



(a)



(b)

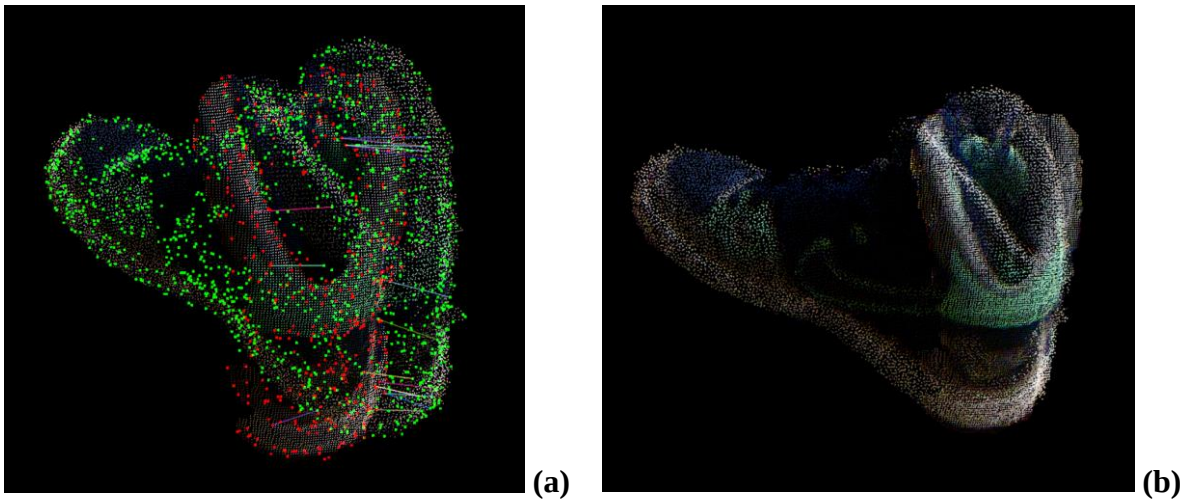
Figura 72: Teste do objeto 3 – Erro na estimativa das correspondências (a) e no alinhamento das nuvens de pontos após 12 iterações (b)

Os resultados desta experiência provaram que as dificuldades obtidas anteriormente não se devem à dimensão e características geométricas do objeto. Assim sendo, nas primeiras iterações do processo de registo não há qualquer erro associado ao processo de alinhamento de nuvens de ponto. Apesar de em algumas situações as correspondências estimadas não serem as mais adequadas, o algoritmo de refinamento consegue resolver estes problemas obtidos pelo método de alinhamento inicial implementado. Contudo, com o avançar deste processo começa a surgir um pequeno erro no alinhamento que se vai propagando às nuvens de pontos seguintes, sendo o método de refinamento incapaz de corrigir esta dificuldade na sua totalidade. Esta acumulação do erro na transformação afeta a estimativa de correspondências entre o modelo global e as novas nuvens de pontos registadas (*Figura 72-a*), de modo a que o modelo resultante se torne inválido nas etapas seguintes (*Figura 72-b*). Desta forma, foram estudadas diferentes combinações de parâmetros destes algoritmos para encontrar uma forma de resolver estes

problemas para o objeto em questão. No entanto, o aumento do valor dos parâmetros provocou um aumento do tempo de processamento do registo de nuvens de pontos.

De seguida, foi avaliada a influência do filtro de *downsampling* na diminuição da resolução do modelo resultante à saída do processo de registo. Para isso, foram utilizadas nuvens de pontos com um menor nível de filtro inicial. Contudo, nuvens de pontos com uma elevada quantidade de pontos e com descritores de características iguais ou idênticos originaram problemas de deslizamento de planos, onde certos pontos de determinadas superfícies contribuíram negativamente para a distância métrica global devido à existência de correspondências ambíguas. A solução passou por negligenciar todos os pontos com características que são consideráveis dominantes no conjunto de dados, e assim, concentrar-se mais em pontos proeminentes (salientes). No entanto, os resultados obtidos foram idênticos aos anteriores, tendo o processo de registo sido comprometido na mesma iteração ou em anteriores.

Como a estimativa de correspondências teve grandes dificuldades para iterações mais avançadas, foram equacionados novos algoritmos e métodos quer de estimativa quer de rejeição destes conjuntos. Outros métodos de rejeição de correspondência foram testados, tirando vantagem de informação extra das nuvens de pontos em análise, tais como a informação das normais ou estatística das correspondências, como por exemplo distância média das correspondências. Contudo, a utilização destes métodos teve influência na quantidade de correspondências obtidas após rejeição o que traduziu numa diminuição deste conjunto que possibilitou a ocorrência de falhas nas iterações iniciais. A *Figura 73* ilustra um processo de rejeição de correspondências mais limitado e o resultado obtido no alinhamento através destas relações para esta experiência.



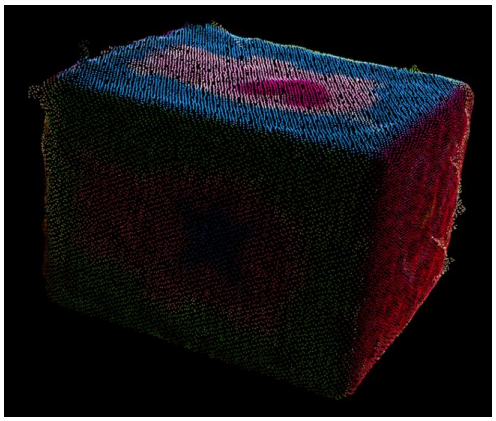
(a) **(b)**

Figura 73: Teste do objeto 4 – Resultado da estimativa das correspondências (a) e do alinhamento das nuvens de pontos após 8 iterações (b)

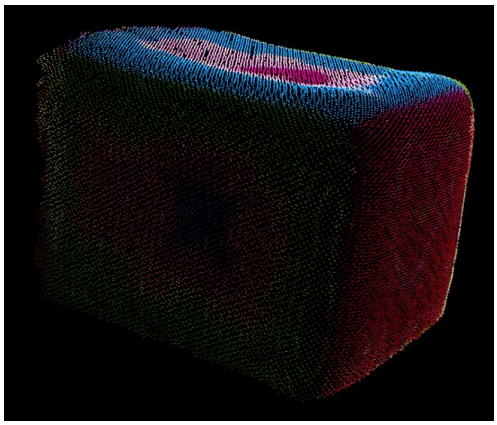
7.10 Processamento do modelo

O algoritmo *MLS* foi escolhido para suavizar o modelo resultante do processo de registo de nuvens de pontos. Devido à natureza ruidosa dos dados adquiridos pelo sensor *Kinect*, este método foi utilizado com o objetivo de garantir um melhor aspeto da malha tridimensional final de cada objeto. Como comparação, na reconstrução de uma malha a partir de um modelo sem este processamento de informação, verificou-se que a definição dos contornos nas extremidades é rígida e as superfícies não são limpas. Porém, o tempo de processamento deste algoritmo comparado com os utilizados anteriormente é muito elevado. O tempo de processamento está relacionado com a densidade de cada nuvem utilizado neste processo. Para tal, à nuvem de pontos que contém o modelo resultante foi aplicado um filtro inicial para redução da resolução, de modo a que o tempo de processamento seja diminuído. Como consequência, a estimativa das normais para esta nuvem de pontos teve de ser recalculada. Por fim, as nuvens de pontos resultantes, ou seja, as nuvens de pontos que foram suavizadas e as normais para cada ponto, são combinadas numa nuvem de pontos contendo 5 dimensões, que inclui: as posições dos pontos nos eixos *x*, *y* e *z*, a informação da cor em formato *RGB* e os dados referentes as normais da superfície.

A configuração dos parâmetros deste algoritmo foi testada manualmente para se encontrar os valores para os quais se encontram os melhores resultados. Assim sendo, a utilização de um polinómio de segundo grau neste algoritmo garante bons resultados, sendo capaz de diminuir algum do erro inerente à captura de informação por este sensor. A influência do raio de estimativa foi estudada de modo a encontrar a melhor relação entre a eliminação de parte do ruído e da manutenção das características geométricas do objeto. A *Figura 74* ilustra os resultados da modificação do parâmetro deste algoritmo para dois valores distintos.



(a)



(b)

Figura 74: Caracterização da influência do raio de estimativa para um valor pequeno (a) e para um valor elevado (b)

Na *Figura 74-a*, é possível visualizar que para um valor pequeno, neste caso específico foi definido como 0,01, as características geométricas do objeto são ligeiramente afetadas, sendo capaz de descrever de forma satisfatória o objeto em questão. Contudo, para este valor a quantidade de ruído presente nos dados é mantida e, desta forma pode-se concluir que para valores pequenos a eliminação do ruído é quase inexistente. Pelo contrário, na *Figura 74-b* devido à utilização de um raio de 0,05, é possível verificar que a presença de ruido é bastante inferior, no entanto as características geométricas do objeto são comprometidas. Desta forma, quanto maior for o valor do raio de estimativa maior será a perda de detalhe da superfície do objeto. Assim sendo, a solução encontrada passou por encontrar um valor intermédio, neste caso este parâmetro foi definido como 0,02.

A utilização dos métodos de dilatação por rede *voxel* e dilatação do plano local de cada ponto permite ajustar a densidade das nuvens de pontos, criando superfícies mais consistentes. Assim sendo, verificou-se que quanto maior for o valor de dilatação (tamanho da rede *voxel*), maior será o afastamento entre os pontos. Após aplicada esta dilatação, o método de dilatação do plano permite redefinir a vizinhança a partir de cada um dos pontos do modelo. Porém, quanto maior for o raio de estimativa maior será a sobreposição dos vizinhos de pontos que se encontrem lado a lado e, como tal a textura da superfície desse objeto será comprometida. Um problema imediato da aplicação isolada deste método é que este adiciona a mesma quantidade de novas amostras para todos os pontos, não tendo em conta a densidade local. Contudo, a utilização da rede *voxel* descrita permite que a densidade do ponto seja uniforme.

A *Tabela 24* descreve os parâmetros que originaram os melhores resultados para os modelos obtidos pela aplicação implementada.

Tabela 24: Caracterização dos parâmetros estabelecidos no algoritmo *MLS*

Função do algoritmo *MLS*	Valor do parâmetro
setSearchRadius	*0.02*
setSqrGaussParam	*0.02 * 0.02*
setPolynomialFit	*true*
setPolynomialOrder	*2*
setUpsamplingMethod	*VOXEL_GRID_DILATION*
setDilationVoxelSize	*0.002*
setDilationIterations	*Não definido*
setUpsamplingMethod	*SAMPLE_LOCAL_PLANE*
setUpsamplingRadius	*0.002*
setUpsamplingStepSize	*0.001*

7.11 Reconstrução do modelo

Para a reconstrução do modelo, foi implementado o algoritmo *Greedy Triangulation* (*GT*). A nuvem de pontos de entrada para este método de triangulação é a nuvem de pontos resultante do processo de processamento através do algoritmo *MLS*. Como mencionado anteriormente, as superfícies são estimadas pela criação de triângulos a partir da informação contida nas nuvens de pontos. Cada triângulo criado representa uma face na malha tridimensional final. Por último, a malha resultante será exportada para um formato que os programas de *CAD* possam ser capazes interpretar.

Para avaliar a qualidade das malhas tridimensionais obtidas por este algoritmo foi estudada a influência dos parâmetros deste algoritmo por forma a alcançar um bom resultado capaz de descrever as características geométricas de um determinado objeto. Devido à falta de fundamentação teórica acerca da definição dos valores típicos e das consequências da modificação dos parâmetros deste algoritmo, alguns dos valores utilizados foram encontrados em tutoriais da biblioteca *PCL*. Contudo, foi necessário ajustar estes parâmetros em relação à densidade da nuvem de ponto. Os parâmetros mais sensíveis deste algoritmo são o raio de procura e o tamanho da vizinhança. Desta forma, foi possível concluir que o aumento do raio de procura permite estabelecer triângulos de dimensões superiores. Contudo, a utilização de maiores triângulos foi capaz de afetar as características geométricas do objeto. Por outro lado, a modificação do valor dos ângulos não provou ter algum efeito inesperado. Para além disso, verificou-se que o tempo necessário para a reconstrução da superfície de um objeto aumenta com a quantidade de informação presente na nuvem de pontos. Porém, a reconstrução para os objetos utilizados não foi superior a *1 min*, apesar de os modelos utilizados corresponderem a resultados parciais do modelo global esperado. A *Tabela 25* descreve os parâmetros que originaram os melhores resultados para os modelos obtidos pela aplicação implementada.

Tabela 25: Caracterização dos parâmetros estabelecidos no algoritmo *GT*

Função do algoritmo *GT*	Valor do parâmetro
setSearchRadius	*0.005*
setMu	*2.5*
setMaximumNearestNeighbors	*1000*
setMaximumSurfaceAngle	*45º*
setNormalConsistency	*false*
setMinimumAngle	*10º*
setMaximumAngle	*120º*

7.12 Discussão dos resultados da reconstrução

A partir dos modelos parciais dos objetos obtidos no processo de registo de nuvens de pontos, foram aplicadas técnicas de processamento e reconstrução de nuvens de pontos para permitir que estes modelos possam ser utilizados em programas de *CAD*. Assim sendo, as malhas tridimensionais finais para cada objeto testado anteriormente são apresentadas nas *Figuras 75*, *76* e *77*, após a aplicação do algoritmo *Moving Least Squares* (*MLS)* utilizado para suavizar a informação e do algoritmo de reconstrução da superfície pelo método *Greedy Triangulation* (*GT*).

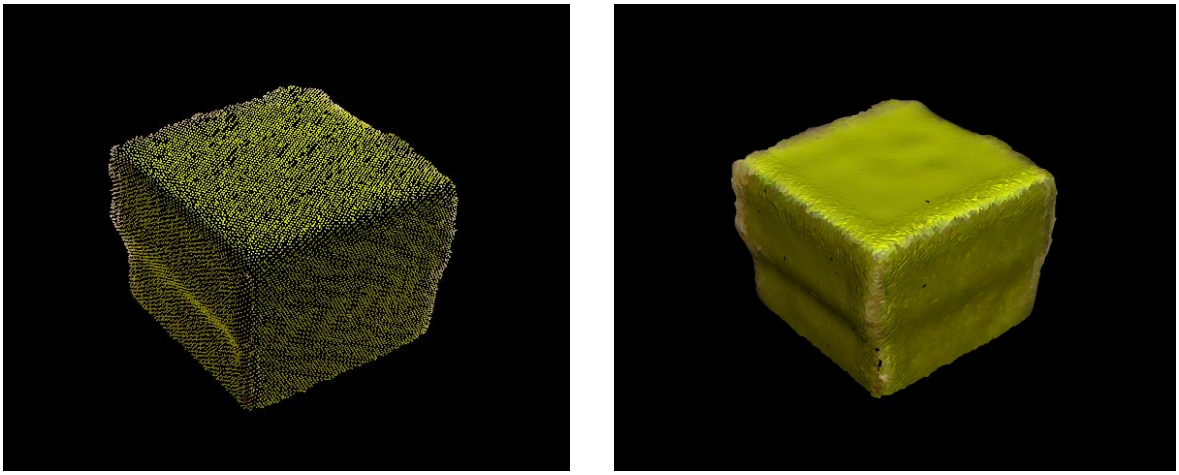


Figura 75: Reconstrução do objeto 2 - Modelo da nuvem de pontos suavizada pelo algoritmo *MLS* (a) Reconstrução da superfície por triangulação pelo método *GT* (b)

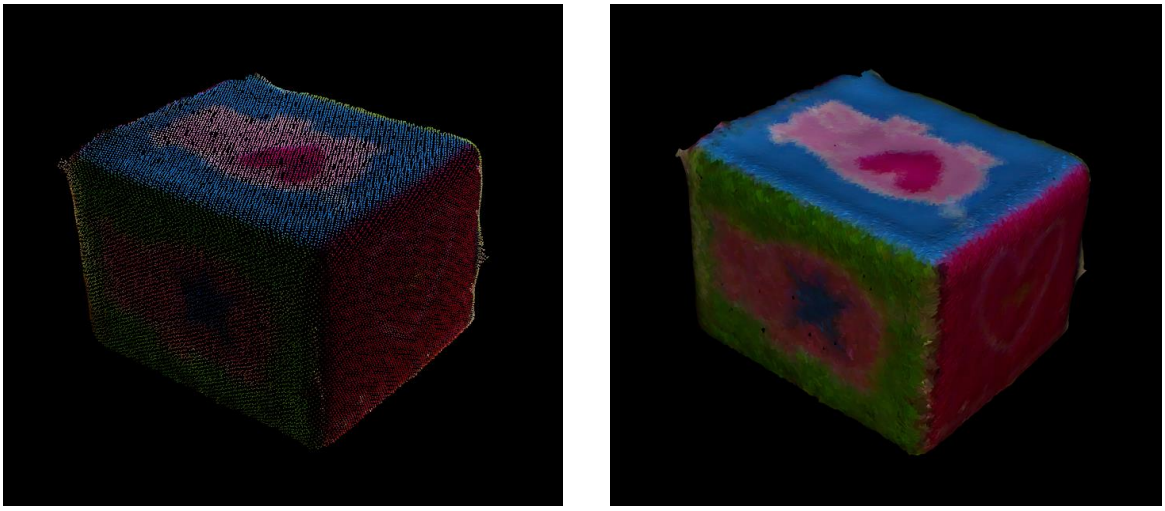


Figura 76: Reconstrução do objeto 3 - Modelo da nuvem de pontos suavizada pelo algoritmo *MLS* (a) Reconstrução da superfície por triangulação pelo método *GT* (b)

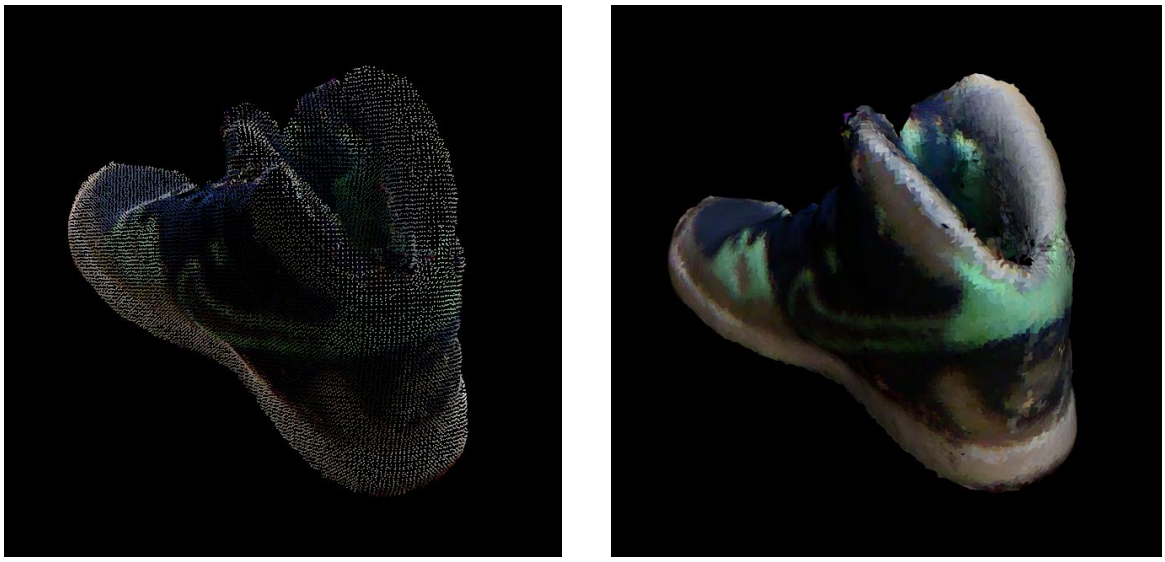


Figura 77: Reconstrução do objeto 4 - Modelo da nuvem de pontos suavizada pelo algoritmo *MLS* (a) Reconstrução da superfície por triangulação pelo método *GT* (b)

Pela análise dos resultados obtidos anteriormente, verificou-se que o algoritmo responsável pela suavização, o *MLS*, permite eliminar algum do ruído presente no modelo sem afetar significativamente as características geométricas do objeto. Pela utilização dos dois métodos contidos neste algoritmo, para além de diminuir o ruido é possível ajustar a densidade do modelo resultante. Assim sendo, com este método é possível definir uma densidade constante em toda a nuvem de pontos, evitando a existência de maiores orifícios na superfície recriada pelo método de triangulação. Desta forma, é visível nas imagens que os resultados deste algoritmo são consistentes e conseguem traduzir corretamente o objeto em questão, embora ainda seja visível um ligeiro ruido nas extremidades dos objetos.

Porém, como o algoritmo *GT* utiliza todos os pontos da nuvem de entrada de forma a criar uma malha tridimensional com base na sua triangulação, a existência de ruído ou regiões que tenham sido mal registadas pode originar pequenos defeitos na superfície do objeto. Tal como referido anteriormente, a utilização do algoritmo *MLS* permite reduzir esta dificuldade. Na *Figura 75-b* e *Figura 76-b* pode ser visto a existência de pequenos orifícios na superfície do objeto e de uma reconstrução mais imprecisa nos contornos do objeto. Na *Figura 77-b*, é visível que a reconstrução de objetos com maior detalhe é vagamente comprometida em certas regiões da superfície. Contudo, os resultados da reconstrução de objetos por este método triangulação embora não sejam ideais são satisfatórios. O algoritmo *Marching Cubes* (*MC*) referido na fundamentação teórica também foi experimentado, no entanto não foi possível obter-se bons resultados por este método.

7.13 Discussão global dos resultados

Após a avaliação dos diversos algoritmos contidos na biblioteca *PCL* para o processamento e reconstrução *3D* de nuvens de pontos verificou-se que estes permitem a sua integração em aplicações para digitalização de objetos. O trabalho realizado nesta dissertação é o primeiro passo dado nesse sentido, sendo que alguns dos algoritmos aplicados neste sistema de digitalização podem ser também utilizados em outras aplicações que requerem a informação tridimensional de objetos e de cenas envolventes.

A aquisição dos dados adquiridos pelo sensor *Kinect*, que por natureza são bastante ruidosos e propícios a erros de medição, influenciam os resultados no registo de nuvens de pontos. Assim sendo, verificou-se a existência de algumas dificuldades quando os objetos se posicionavam a uma distância superior do sensor, que por sua vez afetaram a estimativa das correspondências e a descrição das características locais da sua superfície. Desta forma, a digitalização dos objetos tem de ser realizada o mais perto possível do sensor, tendo em conta as especificações do mesmo. Para além disso, a utilização de métodos de filtragem e segmentação permite reduzir o tempo de processamento deste sistema, devido a eliminação de algum do ruído e da informação desnecessária da cena envolvente. Estas técnicas possibilitam uma maior eficácia no processamento dos dados tridimensionais, pois apenas a informação dos objetos em questão são utilizadas, e ainda uma diminuição da probabilidade de ocorrência de erros nestas etapas.

Nas etapas de processamento, é necessário fazer um ajuste dos parâmetros dos diversos algoritmos, tendo em conta a resolução das nuvens de pontos utilizadas. Os resultados obtidos em termos qualitativos após esta operação são bastante recomendáveis, tenho em conta a informação *3D* das nuvens de pontos provenientes do sensor *Kinect*.

A correta estimativa das características dos objetos nas diversas nuvens de pontos tem uma grande influência nos resultados obtidos. Assim sendo, um bom alinhamento inicial é alcançado se forem estimadas boas correspondências entre os pontos de interesse detetados em ambas as nuvens de pontos. A partir dos resultados encontrados, verificou-se a existência de dificuldades na obtenção de boas correspondências nos casos mais avançados do processo de registo. Esta situação deve-se a uma maior dificuldade na obtenção de boas descrições geométricas e, como consequência algumas falsas correspondências foram produzidas. Contudo, apesar de o alinhamento inicial encontrado não ser o mais recomendável em algumas situações, o *ICP* consegue encontrar a melhor transformação possível. Porém, nem sempre se verifica esta situação. A estimativa da transformação encontrada pelo método *ICP* pode em

alguns casos ficar retido num mínimo local, tendo consequências indesejáveis no processo de registo. Este problema foi uma das principais dificuldades encontradas nesta aplicação.

O método de registo implementado nesta dissertação consiste na concatenação incremental das nuvens de pontos adquiridas. Embora esta implementação seja capaz de construir boas representações dos objetos do mundo real quando a quantidade de capturas é relativamente baixa, ele ainda tem problemas quando o número de capturas começa a aumentar. No processo de registo, o método desenvolvido não consegue superar o erro de transformação acumulado entre as sucessivas nuvens de pontos capturadas e o modelo global resultante. Desta forma, à medida que o número de iterações começa a aumentar, as características geométricas do objeto começam a ser comprometidas. Além destes pequenos erros propagados em cada iteração, a estimativa incorreta de correspondências nas etapas mais avançadas do processo resultam na falha do modelo final. Este problema poderia ser superado se o método de registo incremental fosse mudado para um método baseado em pares de nuvens, em que o processo de registo de nuvens de pontos seja feito apenas com a nuvem previamente capturada. Se for conhecido o número de nuvens de pontos adquiridas pelo sensor e a posição de cada nuvem, a sua transformação pode ser estabelecida a partir da combinação de todas as transformações estimadas até esse ciclo de registo. Contudo, esta abordagem exige um sistema de ajuste em ciclo fechado para compensar o erro de alinhamento acumulado entre as nuvens de pontos.

Depois de terminado o registo de nuvens de pontos, procede-se à suavização das superfícies representadas no modelo resultante através do método *MLS*, para que pequenos erros de medida ou ruído bem como a existência de áreas sobrepostas provenientes do processo de registo sejam minimizadas. Apesar dos problemas referidos terem comprometido o processo de registo, foi avaliado o processamento e reconstrução *3D* dos modelos obtidos até então, ou seja em modelos parciais do esperado deste processo. Desta forma, foram obtidos bons resultados para o algoritmo de suavização, sendo possível ajustar a densidade das nuvens de pontos e eliminar parte do ruído existente sem afetar as características geométricas dos objetos. Assim sendo, o modelo já se encontra nas melhores condições para a reconstrução por triangulação de uma malha tridimensional pelo algoritmo *GT*. Embora existam pequenos orifícios e imprecisões na malha tridimensional gerada por este método, os resultados são considerados satisfatórios. Por fim, a malha tridimensional resultante é exportada para os formatos de ficheiro *.ply*, *.stl* e *.obj* para poderem ser incorporados em programas de *CAD*.

8. CONCLUSÃO E TRABALHO FUTURO

Neste capítulo são descritas as conclusões deste projeto de investigação, sendo salientada a avaliação dos resultados obtidos e o cumprimento dos objetivos propostos nesta dissertação. Por último, como enquadramento futuro as possíveis melhorias são enumeradas no contexto deste projeto.

8.1 Conclusão

As áreas de aplicação dos *scanners 3D* são bastante alargadas, desde arquitetura, engenharia inversa e inspeção de peças, computação gráfica, preservação de artefactos históricos, medicina, entre outras. Embora exista uma vasta aplicabilidade dos *scanners* a *3D* nos dias de hoje, o custo desta tecnologia ainda não é o mais acessível para a utilização pessoal. O trabalho realizado no contexto deste projeto de investigação consistia na avaliação da tecnologia mais acessível e prática para digitalização de objetos de pequenas dimensões. As tecnologias utilizadas para este efeito têm uma diversidade de soluções, que vão desde soluções sem contacto a soluções de contacto passivas e ativas. Para cada uma delas, foram avaliadas as vantagens e desvantagens das técnicas utilizadas para a digitalização de objetos. Através deste estudo, conclui-se que nenhuma tecnologia é perfeita nem capaz de resolver todos os problemas de igual forma. Assim sendo, verificou-se que a tecnologia utilizada para construir um *scanner* a *3D* depende essencialmente da finalidade para o qual foi criada.

Nos últimos anos, a grande evolução tecnológica proporcionou o desenvolvimento de sistemas de visão artificial. Como consequência, a empresa *Microsoft* lançou no mercado o sensor de movimento *Kinect* desenvolvido para facilitar a interação entre jogadores e os jogos eletrónicos através da utilização de gestos e comandos de voz. No entanto, o impacto do lançamento deste sensor estendeu-se à engenharia eletrónica e robótica para desenvolvimento de novas ferramentas de reconhecimento e reconstrução de modelos tridimensionais. Desta forma, a utilização deste dispositivo foi testada neste projeto de investigação. Assim sendo, para facilitar a obtenção dos dados tridimensionais a partir do sensor Kinect foram utilizados os drivers desenvolvidos pela *PrimeSense* e o *framework OpenNI*. A utilização da biblioteca *PCL* foi avaliada para a reconstrução tridimensional dos objetos a partir das nuvens de pontos adquiridas pelo sensor. Desta forma, verificou-se que esta biblioteca possui diversas ferramentas que efetuam o processamento de nuvens de pontos. A partir do estudo dos diversos

módulos desta biblioteca foram utilizados os métodos que melhor responderam ao objetivo proposto. No seguimento deste estudo, foi implementado um sistema para o registo de nuvens de pontos num modelo global capaz de representar o objeto capturado. Esta aplicação foi estudada para vários objetos de dimensões, texturas e características diversificadas. Como tal, para facilitar a digitalização dos objetos foi construído uma plataforma rotativa. Esta plataforma permite que ângulo entre as diferentes capturas seja constante.

Os dados adquiridos a partir do sensor Kinect são de baixa resolução e por natureza bastante ruidosos, contudo este dispositivo representa uma solução atraente comparado com as outras câmaras de profundidade disponíveis no mercado. Ainda assim, a qualidade dos dados é suficiente para a captura e reconstrução de objetos em ambientes *indoor*. Do estudo efetuado, verificou-se que para distâncias pequenas dentro do seu campo de operação, há uma maior dificuldade na digitalização de objetos devido a pequenos orifícios nas nuvens de pontos. O melhor resultado foi obtido com o sensor ligeiramente afastado do valor mínimo e, por forma a capturar o topo da superfície do objeto, a uma altura superior em relação ao ambiente. A utilização de um filtro *Passthrough* foi testada por forma a reduzir alguma da informação da cena envolvente desnecessária para a digitalização de objetos. Assim sendo, a aplicação deste filtro a cada um dos eixos coordenados permitiu definir uma região de interesse no qual apenas será digitalizado o objeto e parte da plataforma rotativa.

Da caracterização da segmentação de planos, conclui-se que o tempo de processamento do algoritmo de segmentação de planos e de extração de *clusters* é reduzido ao ponto de ser aplicado em tempo real. Contudo, após a configuração dos melhores parâmetros, é percetível a existência de pequenos pontos do plano nas extremidades dos objetos segmentados pela aplicação. Estes pequenos erros podem ser reduzidos através do algoritmo de redução de *outliers* aplicado na etapa de pré-processamento das nuvens de pontos. Para além disso, foi verificada a influência das condições de luminosidade e das dimensões dos objetos neste processo. Desta forma, verificou-se que as condições de luminosidade não comprometem a segmentação e, que a diversidade das características geométricas dos objetos podem resultar em más segmentações.

Em relação à estimativa das normais, inferiu-se que o raio de estimativa está relacionado com a densidade da nuvem de pontos. No entanto, este valor não pode ser nem muito baixo nem muito alto, caso contrário a obtenção de normais da superfície distorcidas é bastante provável.

No que respeita à deteção de pontos de interesse, o algoritmo *SIFT* obtém resultados bons e consistentes. A escolha correta dos parâmetros tem uma grande influência na quantidade

de pontos de interesse obtidos, sendo o valor de contraste mínimo muito sensível. Este valor tem de ser encontrado por forma a que os pontos de interesse se encontrem maioritariamente nas características salientes de determinado objeto.

Do estudo dos descritores de características pelo algoritmo *FPFH* deduziu-se que o raio de estimativa influencia a quantidade de correspondências encontradas na etapa seguinte. Para valores superiores também é largamente afetado o tempo de processamento. Para além disso, tem de se garantir que o raio utilizado para deteção dos descritores das características seja superior ao raio utilizado para estimar as normais da superfície, caso contrário podem ocorrer erros na etapa de processamento.

Da caracterização da estimativa e rejeição de correspondências, concluiu-se que para valores muito superiores e inferiores é muito provável a existência de erros de correspondência. Em relação à estimativa de correspondências, dois métodos podem ser utilizados, método direto e reciproco. O método reciproco embora consiga limitar a quantidade de correspondências encontradas, nem sempre esse número é suficiente para estimar corretamente a transformação. Por fim, para ângulos de desfasamento entre nuvens de pontos superiores a 20^{o} não é possível encontrar as relações entre os dois conjuntos.

Do estudo dos métodos de alinhamento inicial, pode-se verificar que a transformação encontrada baseada no algoritmo *SVD* consegue obter melhores resultados para poucas iterações. No caso, do algoritmo *LM* os resultados alcançados nas diversas iterações foram melhores que no algoritmo anterior.

Na estimativa da transformação pelo método *ICP*, verificou-se que em muitos casos, a solução obtida não ter sido a melhor. Isto deve-se ao fato de o *ICP* ter ficado retido num mínimo local, tendo consequências graves no processo de registo. Este fato tornou-se na maior dificuldade encontrada nesta aplicação.

A caracterização do método *MLS* para execução de tarefas de suavização das superfícies amostradas, permitiu inferir a sua utilidade na remoção de erros inerentes e na definição de uma densidade constante às nuvens de pontos capturadas pela *Kinect*. No entanto, o tempo de execução deste método é ainda elevado se não for implementado um filtro de redução da resolução ao modelo que será suavizado.

Por fim, a reconstrução pelo *GT* permite a obtenção de malhas tridimensionais com ligeiras imprecisões e pequenos orifícios, no entanto de uma maneira geral os resultados podem ser considerados satisfatórios.

Desta forma, o *Scanner 3D* proposto nesta dissertação não é o mais preciso nem o digitalizador mais rápido existente, mas talvez seja a tecnologia mais acessível para a digitalização de objetos. O nível de detalhe dos modelos obtidos é suficiente para garantir a correta perceção da geometria dos objetos. Nesta dissertação, foi implementada um arquitetura de sistema associado ao registo de nuvens de pontos num modelo global capaz de representa-lo. Esta aplicação foi estudada para vários objetos de dimensões, texturas e características diversificadas. Contudo, no processo de registo de nuvens de pontos, o método implementado não consegue resolver a acumulação de erro de transformação entre as sucessivas nuvens de pontos causando a falha na operação de construção do modelo global. Assim sendo, à medida que o modelo global resultante começa a crescer, as características geométricas do objeto começam a comprometer o resultado final. Desta forma, o registo de nuvens de pontos num único modelo global capaz de representar na totalidade um determinado objeto não foi conseguido com sucesso. Embora tenham sido encontrados bons resultados para cada um dos algoritmos implementados nesta aplicação, este processo contém ainda alguns problemas. Um dos maiores problemas encontrados foi a deteção de boas correspondências entre os pontos de interesse de ambos os conjuntos nas iterações mais avançadas do processo de registo. Os erros encontrados devido à aquisição dos dados pelo sensor *Kinect* tem uma grande influência nas correspondências dos pontos de interesse. Nas iterações em que não se conseguiram obter correspondências corretas foram originados alguns erros de estimativa do alinhamento inicial que foram transferidos às nuvens de pontos seguintes, que resultaram na falha dessas iterações prejudicando o modelo resultante obtido até esse momento. Apesar destas falhas no processo de registo foram avaliados a reconstrução parcial destes modelos, tendo sido obtidas malhas tridimensionais satisfatórias. Estas malhas foram exportadas para formatos capazes de serem utilizados em programas de *CAD*, sendo corretamente visualizadas nesses sistemas.

8.2 Trabalho Futuro

O trabalho realizado no âmbito desta dissertação permite fazer uma avaliação dos resultados obtidos pela digitalização de objetos a partir de sensores de profundidade de baixo custo. O próximo passo consiste na utilização de outras tecnologias capazes de adquirir os dados tridimensionais de melhor forma, por exemplo através de sensores de profundidade mais potentes e eficientes. Uma das soluções pode passar pela utilização do novo sensor *Kinect 2.0*. Este novo dispositivo permitirá uma maior precisão nos resultados, devido à superior resolução em comparação com o seu antecessor.

No âmbito da aplicação implementada, seria importante avaliar novos métodos de segmentação de planos e extração de objetos. Algumas ferramentas presentes na biblioteca, como os algoritmos *Region Growing Segmentation e Difference of Normals Based Segmentation*, podem ser estudadas por forma a avaliar a sua utilização neste sistema.

Em relação ao registo de nuvens de pontos poderão ser explorados novas metodologias e estudados algoritmos capazes da otimização dos resultados em cada iteração deste processo. Desta forma, será possível reduzir os erros de transformação entre nuvens de pontos na construção do um modelo global de determinado objeto. Para além disso, poderão ser explorados novos algoritmos de deteção de pontos de interesse, como por exemplo o *NARF Keypoints*, e de descrição das características geométricas de objetos, tal como o *SHOT Estimation*, presentes na biblioteca *PCL*.

No que diz respeito à reconstrução dos modelos originados poderá ser avaliada a utilização de novos métodos, que por escassez de tempo não conseguiram ser analisadas e implementadas neste projeto de investigação.

Por fim, seria útil o desenvolvimento de um interface capaz de modificar os valores dos diversos parâmetros utilizados nesta aplicação. Esta abordagem permitiria fazer o ajuste destes parâmetros à digitalização de objetos de diferentes dimensões.

BIBLIOGRAFIA

[1] Yu, F., Lu, Z., Luo, H., and Wang, P., "Three-Dimensional Model Analysis and Processing," *Springer*, 2010.

[2] J. Pelovitz, "What's the Right 3D Scanner for You?," *21 Abril 2014*. [Online]. Available: http://home.lagoa.com/2014/04/whats-the-right-3d-scanner-for-you/.

[3] M. Botsch, "Geometric Modeling: 3D Scanning." [Online]. Available: http://wwwhomes.uni-bielefeld.de/ggoetze/Master/02_Scanning.pdf.

[4] "Scanning Probes - REVO System." [Online]. Available: http://www.cmmxyz.com/revo-system.html.

[5] "Contact vs. Noncontact Measurement for Computer-Aided Inspection," *4 Junho 2013*. [Online]. Available: http://www.qualitymag.com/articles/91150-contact-vs-noncontact-measurement-for-computer-aided-inspection.

[6] "3D Scanners Information." [Online]. Available: http://www.globalspec.com/learnmore/manufacturing_process_equipment/inspection_tools_instruments/3d_scanners.

[7] "National Research Council Canada." [Online]. Available: www.nrc-cnrc.gc.ca/eng/.

[8] R. Mayer, *Scientific Canadian: Invention and Innovation From Canada's National Research Council*. Vancouver: Raincoast Books, 1999.

[9] F. Bernardini and H. Rushmeier, "The 3D model acquisition pipeline," *Computer Graphics Forum*, vol. 21. pp. 149–172, 2002.

[10] "3d Scanning Technology And 3d Modeling Information Technology Essay." [Online]. Available: http://www.ukessays.com/essays/information-technology/3d-scanning-technology-and-3d-modeling-information-technology-essay.php.

[11] E. Rakitin, I. Rakitin, V. Staleva, F. Arnaoutoglou, A. Koutsoudis, and G. Pavlidis, "An overview of 3D laser scanning technology." [Online]. Available: http://www.ipet.gr/~akoutsou/docs/AO3D.pdf.

[12] A. Peiravi and B. Taabbodi, "A Reliable 3D Laser Triangulation-based Scanner with a New Simple but Accurate Procedure for Finding Scanner Parameters," *J. Am. Sci.*, vol. 6, pp. 80–85, 2010.

[13] W. Böhler and A. Marbs, "3D scanning instruments," in *Proceedings of the CIPA WG 6 International Workshop on Scanning for Cultural Heritage Recording*, 2002, pp. 9–18.

[14] P. C. Dawson, R. M. Levy, and G. Mackay, “Documenting Mackenzie Inuit architecture using 3D laser scanning,” *Alaska J. Anthropol.*, vol. 7, pp. 29–44, 2009.

[15] J. Davis and X. Chen, “A laser range scanner designed for minimum calibration complexity,” *Proc. Third Int. Conf. 3-D Digit. Imaging Model.*, pp. 91–98, 2001.

[16] G. Pavlidis, A. Koutsoudis, F. Arnaoutoglou, V. Tsioukas, and C. Chamzas, “Methods for 3D digitization of Cultural Heritage,” *J. Cult. Herit.*, vol. 8, pp. 93–98, 2007.

[17] J. Geng, “Structured-light 3 D surface imaging: a tutorial,” *Adv. Opt. Photonics*, vol. 3, p. 128, 2011.

[18] C. Rocchini, P. Cignoni, C. Montani, P. Pingi, and R. Scopigno, “A low cost 3D scanner based on structured light,” *Comput. Graph. Forum*, vol. 20, pp. 299–308, 2001.

[19] “Structured Light vs Microsoft Kinect,” *26 Fevereiro 2012*. [Online]. Available: http://www.hackengineer.com/structured-light-vs-microsoft-kinect/.

[20] M. Morimoto and K. Fujii, “A portable 3D scanner based on structured light and stereo camera,” in *ISCIT 2005 - International Symposium on Communications and Information Technologies 2005, Proceedings*, 2005, vol. II, pp. 550–553.

[21] I. Stančić, J. Musić, and V. Zanchi, “Improved structured light 3D scanner with application to anthropometric parameter estimation,” *Meas. J. Int. Meas. Confed.*, vol. 46, pp. 716–726, 2013.

[22] J. Geng, “Optical Imaging Techniques and Applications.” 2011 AAPM Annual Meeting, Vancouver, 2011.

[23] S. Jecić and N. Drvar, “THE ASSESSMENT OF STRUCTURED LIGHT AND LASER SCANNING METHODS IN 3D SHAPE MEASUREMENTS,” in *4th International Congress of Croatian Society of Mechanics*, 2003.

[24] S. Clarkson, “Science Behind 3D Vision.” [Online]. Available: http://www.depthbiomechanics.co.uk/?p=102.

[25] “Optical 3D Scanning: Laser Beam or Structured Light?,” *19 de Junho de 2012*. [Online]. Available: http://blog.shapegrabber.com/2012/06/optical-3d-scanning-laser-beam-or-structured-light.html.

[26] R. Horaud, “Three-Dimensional Sensors Lecture 3: Time of Flight Cameras (Continuous Wave Modulation).” [Online]. Available: http://perception.inrialpes.fr/~Horaud/Courses/pdf/Horaud_3DS_3.pdf.

[27] G. Bradshaw, “Non-Contact Surface Geometry Measurement Techniques,” *Camera*, pp. 1–26, 1999.

[28] “Remote Sensing: Lidar Overview.” [Online]. Available: http://www.rocksense.ca/Research/LiDARTechnology.html.

[29] L. Li, “Time-of-Flight Camera – An Introduction.” 2014.

[30] Geomagic, “3D Scanners - A guide to 3D scanner technology.” [Online]. Available: http://www.rapidform.com/3d-scanners/.

[31] M. Reynolds, J. Doboš, L. Peel, T. Weyrich, and G. J. Brostow, “Capturing Time-of-Flight data with confidence,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2011, pp. 945–952.

[32] A. Mansouri, A. Lathuilièere, F. S. Marzani, Y. Voisin, and P. Gouton, “Toward a 3D multispectral scanner: An application to multimedia,” *IEEE Multimed.*, vol. 14, pp. 40–47, 2007.

[33] National Instruments, “Sistemas de imagens 3D com NI LabVIEW,” *06 Dezembro 2012*. [Online]. Available: http://www.ni.com/white-paper/14103/pt/.

[34] “HALCON’s Imaging Software technique ‘Photometric Stereo’ explained,” *2 Abril 2014*. [Online]. Available: http://multipix.com/support-articles/imaging-software-technique-photometric-stereo-explained/.

[35] C. Hernandez, G. Vogiatzis, and R. Cipolla, “Multiview photometric stereo,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, pp. 548–554, 2008.

[36] R. Woodman, “A Photometric Stereo Approach to Face Recognition,” University of the West of England, Bristol, 2007.

[37] C. Hernández, G. Vogiatzis, and R. Cipolla, “Shadows in three-source photometric stereo,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2008, vol. 5302 LNCS, pp. 290–303.

[38] Z. Zhang, “Microsoft kinect sensor and its effect,” *IEEE Multimedia*, vol. 19. pp. 4–10, 2012.

[39] K. D. Mankoff and T. A. Russo, “The Kinect: A low-cost, high-resolution, short-range 3D camera,” *Earth Surf. Process. Landforms*, vol. 38, pp. 926–936, 2013.

[40] A. Bainbridge-smith and H.-H. Wu, “Advantages of using a Kinect Camera in various applications.”

[41] E. Alecrim, “Além do Xbox: Microsoft lança Kinect para Windows,” *01 de Fevereiro de 2012*. [Online]. Available: http://www.infowester.com/noticias/alem-do-xbox-microsoft-lanca-kinect-para-windows/.

[42] 3D Focus, “Microsoft launching Kinect based 3D scanner called Fusion,” *07 de Março de 2013*. [Online]. Available: http://www.3dfocus.co.uk/3d-news-2/3d-printing-3d-news-2/microsoft-launching-kinect-based-3d-scanner-called-fusion/12436.

[43] J. C. . K. Chow, K. D. Ang, D. D. Lichti, and W. F. Teskey, “PERFORMANCE ANALYSIS OF A LOW-COST TRIANGULATION-BASED 3D CAMERA:

MICROSOFT KINECT SYSTEM," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XXXIX–B5. pp. 175–180, 2012.

[44] K. Khoshelham and S. O. Elberink, "Accuracy and resolution of kinect depth data for indoor mapping applications," *Sensors*, vol. 12, pp. 1437–1454, 2012.

[45] Microsoft, "Kinect for Windows Sensor Components and Specifications." [Online]. Available: http://msdn.microsoft.com/en-us/library/jj131033.aspx.

[46] Mitutoyo U.S.A., "Crysta-Apex S 500/700/900/1200 Series-191- Standard CNC CMM - Product Information." [Online]. Available: http://ecatalog.mitutoyo.com/Crysta-Apex-S-5007009001200-Series-191-Standard-CNC-CMM-C1812.aspx.

[47] Mitutoyo U.S.A., "CRYSTA-Apex S Series - Literature." [Online]. Available: http://www.mitutoyo.com/Images/003/315/2097_CRYSTA_ApexS.pdf.

[48] Mitutoyo U.S.A., "Coordinate Measuring Machines - Experience and Innovation." [Online]. Available: http://www2.mitutoyo.de/uploads/media/Coodinate_measuring_machines_PRE.pdf.

[49] FARO, "FARO Gage & Gage-PLUS." [Online]. Available: http://www.dirdim.com/pdfs/DDI_FARO_Gage.pdf.

[50] Steinsvik Production, "Production of Mechanical Components - Faro Gage Plus." [Online]. Available: http://www.sproduction.no/faro-gage-plus.

[51] L. Ward, "The 10 Breakthrough Products of 2013," *Popular Mechanics*, 2013. [Online]. Available: http://www.popularmechanics.com/technology/gadgets/news/the-10-breakthrough-products-of-2013#slide-6.

[52] MakerBot, "MakerBot Digitizer." [Online]. Available: https://store.makerbot.com/digitizer.

[53] MakerBot, "Digitizer Desktop 3D Scanner - User Manual." [Online]. Available: https://s3.amazonaws.com/downloads-makerbot-com/digitizer/MakerBotDigitizer_UserManual.pdf.

[54] STT Engineering & Systems, "FACESCAN - Structured Light face scanner." [Online]. Available: http://www.stt-systems.com/en/products/scanners/face-scanner/face-scanner/.

[55] DAVID Vision Systems, "DAVID Structured Light Scanner SLS-2." [Online]. Available: https://ssl.david-vision-systems.de/shop/product_info.php/language/en/info/p129_DAVID-Structured-Light-Scanner-SLS-2.html.

[56] DAVID Vision Systems, "David LaserScanner." [Online]. Available: http://www.david-3d.com/.

[57] DAVID Vision Systems, “DAVID-SLS-2 Product Launch - News.” [Online]. Available: http://www.david-3d.com/?section=News.

[58] Leica Geosystems, “Leica Geosystems - Leica ScanStation C10.” [Online]. Available: http://hds.leica-geosystems.com/en/Leica-ScanStation-C10_79411.htm.

[59] Leica Geosystems, “Leica ScanStation C10 Product Specifications.” [Online]. Available: http://hds.leica-geosystems.com/downloads123/hds/hds/ScanStation C10/brochures-datasheet/Leica_ScanStation_C10_DS_en.pdf.

[60] Fuel 3D Technologies, “Fuel3D Product.” [Online]. Available: http://uk.fuel-3d.com/product/.

[61] Fuel 3D Technologies, “A technical overview of the Fuel3D system.” [Online]. Available: https://uk.fuel-3d.com/wp-content/uploads/2014/08/FUEL3D-Tech-Paper-A4.pdf.

[62] Google, “Project Tango.” [Online]. Available: https://www.google.com/atap/projecttango/#project.

[63] Occipital Inc., “Structure Sensor Store.” [Online]. Available: https://store.structure.io/store.

[64] KickStarter, “Structure Sensor: Capture the World in 3D.” [Online]. Available: https://www.kickstarter.com/projects/occipital/structure-sensor-capture-the-world-in-3d.

[65] Occipital Inc., “Occipital’s Structure Sensor.” [Online]. Available: http://www.xconomy.com/boulder-denver/2013/09/17/occipital-debuts-mobile-friendly-3d-scanner-kickstarter-campaign/attachment/occipitalstructuresensor/.

[66] Skanect, “Skanect - Features.” [Online]. Available: http://skanect.occipital.com/features/.

[67] cd-soft, “Skanect - 3D Scanning Software.” [Online]. Available: http://cdsoft.com.au/p/8734181/skanect---3d-scanning-software.html.

[68] SolidSmack, “Skanect is the Newest in New Power Scanning Via Kinect.” [Online]. Available: http://www.solidsmack.com/fabrication/skanect-is-the-newest-in-new-power-scanning-via-kinect/.

[69] ThingLab, “Artec Studio 9 + Kinect.” [Online]. Available: http://thinglab.com.au/scan/3d-scanners/artec-studio9-kinect.

[70] Artec 3D Scanners, “Artec Studio 9.2.” [Online]. Available: http://www.artec3d.com/software/studio-kinect/.

[71] Artec 3D Scanners, “Announcing release of Artec Studio 9.0.” [Online]. Available: http://www.artec3d.com/news/Announcing+release+of+Artec+Studio+9.0_4096.

[72] L. Cruz, D. Lucio, and L. Velho, “Kinect and RGBD images: Challenges and applications,” in *Proceedings: 25th SIBGRAPI - Conference on Graphics, Patterns and Images Tutorials, SIBGRAPI-T 2012*, 2012, pp. 36–49.

[73] E. Enaloo, “PrimeSense confirme as Project Natal hardware source.” [Online]. Available: http://www.ehsanenaloo.com/primesense-confirmed-as-project-natal-hardware-source/#more-768.

[74] Robotica Unileon Group, “PCL/OpenNI tutorial 1: Installing and testing.” [Online]. Available: http://robotica.unileon.es/mediawiki/index.php/PCL/OpenNI_tutorial_1:_Installing_and_testing.

[75] J. Martínez, “Instalación OpenNI, Sensor Kinect y NITE en GNU/Linux Ubuntu 11.10 desde fuentes.” [Online]. Available: http://blog.jorgeivanmeza.com/2011/12/instalacion-openni-sensor-kinect-y-nite-en-gnulinux-ubuntu-11-10-desde-fuentes/.

[76] Slash Gear, “Apple, PrimeSense motion-tracking tech company deal confirmed.” [Online]. Available: http://www.slashgear.com/apple-primesense-motion-tracking-tech-company-deal-confirmed-24306674/.

[77] Point Cloud Library, “Point Cloud Library: About.” [Online]. Available: http://pointclouds.org/about/.

[78] Point Cloud Library, “Point Cloud Library: Module common.” [Online]. Available: http://docs.pointclouds.org/1.7.0/group__common.html.

[79] Point Cloud Library, “Point Cloud Library: Module features.” [Online]. Available: http://docs.pointclouds.org/trunk/group__features.html.

[80] Point Cloud Library, “Point Cloud Library: Module filters.” [Online]. Available: http://docs.pointclouds.org/trunk/group__filters.html.

[81] Point Cloud Library, “Point Cloud Library: Module io.” [Online]. Available: http://docs.pointclouds.org/trunk/group__io.html.

[82] Point Cloud Library, “Point Cloud Library: Module kdtree.” [Online]. Available: http://docs.pointclouds.org/trunk/group__kdtree.html.

[83] Point Cloud Library, “Point Cloud Library: Module keypoints.” [Online]. Available: http://docs.pointclouds.org/trunk/group__keypoints.html.

[84] Point Cloud Library, “Point Cloud Library: Module registration.” [Online]. Available: http://docs.pointclouds.org/trunk/group__registration.html.

[85] Point Cloud Library, “Point Cloud Library: Module sample_consensus.” [Online]. Available: http://docs.pointclouds.org/trunk/group__sample__consensus.html.

[86] Point Cloud Library, “Point Cloud Library: Module segmentation.” [Online]. Available: http://docs.pointclouds.org/trunk/group__segmentation.html.

[87] Point Cloud Library, “Point Cloud Library: Module surface.” [Online]. Available: http://docs.pointclouds.org/trunk/group__surface.html.

[88] Point Cloud Library, “Point Cloud Library: Module visualization.” [Online]. Available: http://docs.pointclouds.org/trunk/group__visualization.html.

[89] A.-E. Ichim, “RGB-D Handheld Mapping and Modeling.” [Online]. Available: http://alexichim.com/wp-content/images/Alexandru_Ichim_EPFL_MSC_thesis_final_small.pdf.

[90] Point Cloud Library, “PCL: Compiling from source.” [Online]. Available: http://pointclouds.org/downloads/source.html.

[91] D. Beman and A. David, “Boost C++ Libraries.” [Online]. Available: http://www.boost.org/.

[92] B. Jacob and G. Guennebaud, “Eigen.” [Online]. Available: http://eigen.tuxfamily.org/index.php?title=Main_Page#Credits.

[93] M. Muja and D. G. Lowe, “FLANN - Fast Library for Approximate Nearest Neighbors.” [Online]. Available: http://www.cs.ubc.ca/research/flann/.

[94] Kitware, “VTK - Visualization ToolKit.” [Online]. Available: http://www.vtk.org/.

[95] NVIDIA Comporation, “CUDA.” [Online]. Available: https://developer.nvidia.com/cuda-zone.

[96] QHull, “QHull.” [Online]. Available: http://www.qhull.org/.

[97] J. Hyv, “SURFACE RECONSTRUCTION OF POINT CLOUDS CAPTURED WITH MICROSOFT KINECT,” Oulu University of Applied Sciences, 2012.

[98] AUTODESK, “Point Cloud Object.” [Online]. Available: http://knowledge.autodesk.com/support/3ds-max/learn-explore/caas/documentation/3DSMAX/16/ENU/3ds-Max-Help/files/GUID-49CE0ACB-1345-4D50-B6E5-361DBFDB5B33-htm.html.

[99] Point Cloud Library, “Adding your own custom PointT type.” [Online]. Available: http://pointclouds.org/documentation/tutorials/adding_custom_ptype.php#id1.

[100] Point Cloud Library, “The PCD (Point Cloud Data) file format.” [Online]. Available: http://pointclouds.org/documentation/tutorials/pcd_file_format.php.

[101] Robotica Unileon, “PCL/OpenNI tutorial 2: Cloud processing (basic).” [Online]. Available: http://robotica.unileon.es/mediawiki/index.php/PCL/OpenNI_tutorial_2:_Cloud_processing_(basic).

[102] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, “Towards 3D Point cloud based object maps for household environments,” *Rob. Auton. Syst.*, vol. 56, pp. 927–941, 2008.

[103] Point Cloud Library, “Removing outliers using a StatisticalOutlierRemoval filter.” [Online]. Available: http://pointclouds.org/documentation/tutorials/statistical_outlier.php#statistical-outlier-removal.

[104] Point Cloud Library, “Removing outliers using a Conditional or RadiusOutlier removal.” [Online]. Available: http://pointclouds.org/documentation/tutorials/remove_outliers.php#remove-outliers.

[105] T. Le-tien, M. Luong, T. P. Ho, V. D. Tran, and U. Paris, “3D Reconstruction using Kinect Sensor and Parallel Processing Reconstruction using Kinect Sensor and Parallel Processing on 3D Graphics Processing Unit,” vol. 3, no. 1, pp. 59–67, 2013.

[106] Robotica Unileon, “PCL/OpenNI tutorial 3: Cloud processing (advanced).” [Online]. Available: http://robotica.unileon.es/mediawiki/index.php/PCL/OpenNI_tutorial_3:_Cloud_processing_(advanced)#Model_fitting_.28RANSAC.29.

[107] R. B. Rusu, “Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments,” Institut für Informatik der Technischen Universität München.

[108] Point Cloud Library, “How to use Random Sample Consensus model.” [Online]. Available: http://pointclouds.org/documentation/tutorials/random_sample_consensus.php#random-sample-consensus.

[109] J. Delmerico, “PCL Tutorial: The Point Cloud Library By Example,” 2013. [Online]. Available: http://www.jeffdelmerico.com/wp-content/uploads/2014/03/pcl_tutorial.pdf.

[110] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *Int. J. Comput. Vis.*, vol. 60, pp. 91–110, 2004.

[111] P. Scovanner, S. Ali, and M. Shah, “A 3-dimensional sift descriptor and its application to action recognition,” *15th Int. Conf. Multimed.*, pp. 357–360, 2007.

[112] A. Flint, A. Dick, and A. Van Den Hengel, “Thrift: Local 3D structure recognition,” in *Proceedings - Digital Image Computing Techniques and Applications: 9th Biennial Conference of the Australian Pattern Recognition Society, DICTA 2007*, 2007, pp. 182–188.

[113] “SIFT: Scale Invariant Feature Transform.” [Online]. Available: http://www.maxwell.vrac.puc-rio.br/17050/17050_5.PDF.

[114] C. Harris and M. Stephens, “A Combined Corner and Edge Detector,” *Procedings Alvey Vis. Conf. 1988*, pp. 147–151, 1988.

[115] S. Filipe and L. A. Alexandre, “A Comparative Evaluation of 3D Keypoint Detectors.”

[116] I. Sipiran and B. Bustos, “Harris 3D: A robust extension of the Harris operator for interest point detection on 3D meshes,” in *Visual Computer*, 2011, vol. 27, pp. 963–976.

[117] Point Cloud Library, “Estimating Surface Normals in a PointCloud.” [Online]. Available: http://pointclouds.org/documentation/tutorials/normal_estimation.php.

[118] Point Cloud Library, “Point Feature Histograms (PFH) descriptors.” [Online]. Available: http://pointclouds.org/documentation/tutorials/pfh_estimation.php#pfh-estimation.

[119] R. B. Rusu, N. Blodow, and M. Beetz, “Fast Point Feature Histograms (FPFH) for 3D registration,” *2009 IEEE Int. Conf. Robot. Autom.*, 2009.

[120] Point Cloud Library, “Fast Point Feature Histograms (FPFH) descriptors.” [Online]. Available: http://pointclouds.org/documentation/tutorials/fpfh_estimation.php#fpfh-estimation.

[121] Point Cloud Library, “The PCL Registration API.” [Online]. Available: http://pointclouds.org/documentation/tutorials/registration_api.php#registration-api.

[122] Point Cloud Library, “Point Cloud Library: Transformation Estimation SVD.” [Online]. Available: http://docs.pointclouds.org/1.7.1/classpcl_1_1registration_1_1_transformation_estimation_s_v_d.html.

[123] Point Cloud Library, “Point Cloud Library: Transformation Estimation PointToPlane LLS.” [Online]. Available: http://docs.pointclouds.org/1.7.1/classpcl_1_1registration_1_1_transformation_estimation_point_to_plane_l_l_s.html.

[124] M. Santala, “3D CONTENT CAPTURING AND RECONSTRUCTION USING MICROSOFT KINECT DEPTH CAMERA.” 2012.

[125] Point Cloud Library, “Point Cloud Library: Iterative Closest Point.” [Online]. Available: http://docs.pointclouds.org/1.7.1/classpcl_1_1_iterative_closest_point.html.

[126] Point Cloud Library, “Point Cloud Library: Iterative Closest Point Non Linear.” [Online]. Available: http://docs.pointclouds.org/1.7.1/classpcl_1_1_iterative_closest_point_non_linear.html.

[127] Point Cloud Library, “Point Cloud Library: Iterative Closest Point With Normals.” [Online]. Available: http://docs.pointclouds.org/1.7.1/classpcl_1_1_iterative_closest_point_with_normals.html.

[128] N. Electronics and N. Pathom, “Method of 3D Mesh Reconstruction from Point Cloud U sing Elementary Vector and Geometry Analysis,” pp. 156–159, 2007.

[129] A.-E. Ichim, “PCL - Surface Reconstruction Toyota Code Sprint.” [Online]. Available: http://www.pointclouds.org/assets/icra2012/surface.pdf.

[130] D. Levin, “The approximation power of moving least-squares,” *Mathematics of Computation*, vol. 67. pp. 1517–1532, 1998.

[131] M. T. Dickerson, R. L. Scot Drysdale, S. A. McElfresh, and E. Welzl, “Fast greedy triangulation algorithms,” *Computational Geometry*, vol. 8. pp. 67–86, 1997.

[132] W. E. Lorensen and H. E. Cline, “Marching cubes: A high resolution 3D surface construction algorithm,” *ACM SIGGRAPH Computer Graphics*, vol. 21. pp. 163–169, 1987.

[133] EECS 466 Project, “Point Cloud Reconstruction - Marching Cubes and Poisson.” [Online]. Available: https://sites.google.com/site/pointcloudreconstruction/marching-cubes.

[134] Bot´n Roll, “MOTOR DE PASSO PARA IMPRESSORA 3D (3,5KG).” [Online]. Available: https://www.google.pt/webhp?sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8#q=bot n roll.

[135] Bot´n Roll, “BIG EASY DRIVER.” [Online]. Available: http://www.botnroll.com/product.php?id_product=575.

[136] Arduino, “Arduino Mega 2560,” 2014. [Online]. Available: http://arduino.cc/en/Main/arduinoBoardMega2560.

[137] Atmel Corporation, “Atmel Studio 6,” 2014. [Online]. Available: http://www.atmel.com/microsite/atmel_studio6/.

ANEXO I – PROJETO SOLIDWORKS